UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS CORNÉLIO PROCÓPIO
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

FAGNER CHRISTIAN PAES

DETECÇÃO AUTOMÁTICA DE INCOMPATIBILIDADES CROSS-BROWSER USANDO APRENDIZADO DE MÁQUINA E COMPARAÇÃO DE IMAGENS

DISSERTAÇÃO DE MESTRADO

CORNÉLIO PROCÓPIO

2018

FAGNER CHRISTIAN PAES

DETECÇÃO AUTOMÁTICA DE INCOMPATIBILIDADES CROSS-BROWSER USANDO APRENDIZADO DE MÁQUINA E COMPARAÇÃO DE IMAGENS

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Universidade Tecnológica Federal do Paraná – UTFPR como requisito parcial para a obtenção do título de "Mestre em Informática".

Orientador: Prof. Dr. Willian Massami Watanabe

CORNÉLIO PROCÓPIO

2018

Dados Internacionais de Catalogação na Publicação

P126 Paes, Fagner Christian

Detecção automática de incompatibilidades cross-browser usando aprendizado de máquina e comparação de imagens / Fagner Christian Paes. – 2018.
57 f. : il. color.; 31 cm.

Orientador: Willian Massami Watanabe.
Dissertação (Mestrado) – Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Informática. Cornélio Procópio, 2018.
Bibliografia: p. 54-57.

1. Cross-browser. 2. Browsers (Programa de computador). 3. Aprendizado do computador. 4. Informática – Dissertações. I. Watanabe, Willian Massami, orient. II. Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Informática. III. Título.

CDD (22. ed.) 004

Biblioteca da UTFPR - Câmpus Cornélio Procópio
Bibliotecários/Documentalistas responsáveis:
Simone Fidêncio de Oliveira Guerra – CRB-9/1276
Romeu Righetti de Araujo – CRB-9/1676

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Cornélio Procópio
Programa de Pós-Graduação em Informática

Título da Dissertação Nº 48:

"DETECÇÃO AUTOMÁTICA DE INCOMPATIBILIDADES CROSS-BROWSER USANDO APRENDIZADO DE MÁQUINA E COMPARAÇÃO DE IMAGENS".

por

Fagner Christian Paes

Orientador: **Prof. Dr Willian Massami Watanabe**

Esta dissertação foi apresentada como requisito parcial à obtenção do grau de MESTRE EM INFORMÁTICA – Área de Concentração: Computação Aplicada, pelo Programa de Pós-Graduação em Informática – PPGI – da Universidade Tecnológica Federal do Paraná – UTFPR – Câmpus Cornélio Procópio, às 14h do dia 09 de julho de 2018. O trabalho foi _____ pela Banca Examinadora, composta pelos professores:

_____
Prof. Dr. Willian Massami Watanabe
(Presidente – UTFPR-CP)

_____
Profa. Dra. Érica Ferreira de Souza
(UTFPR-CP)

_____
Prof. Dr. Vinícius Humberto Serapilha Durelli
(UFSJ)
Participação à distância via _____

Visto da coordenação:

_____
Danilo Sipoli Sanches
Coordenador do Programa de Pós-Graduação em Informática
UTFPR Câmpus Cornélio Procópio

A Folha de Aprovação assinada encontra-se na Coordenação do Programa.

AGRADECIMENTOS

A Deus por ter me dado saúde e força para superar as dificuldades. Ao meu orientador Prof. Dr. Willian Massami Watanabe, pela oportunidade de trabalhar ao seu lado, pela confiança, paciência, ensinamentos e amizade construída; a minha esposa Sueli Rodrigues Paes, por ser a incentivadora na superação de meus limites, paciência e amor; aos meus pais e familiares, pelo apoio incondicional; meus colegas de trabalho e amigos pelas relevantes colaborações; e aos professores da UTFPR, pela atenção e conhecimentos que contribuíram para minha formação. Minha sincera gratidão a todos vocês!

RESUMO

PAES, Fagner Chrisitan. DETECÇÃO AUTOMÁTICA DE INCOMPATIBILIDADES CROSS-BROWSER USANDO APRENDIZADO DE MÁQUINA E COMPARAÇÃO DE IMAGENS. 58 f. Dissertação de Mestrado – Programa de Pós-graduação em Informática, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2018.

Contexto: *Cross-Browser Incompatibilities* (XBIs) representam problemas de compatibilidade que podem ser observados ao carregar a mesma aplicação Web em diferentes navegadores. Usuários podem interagir com a Web através de distintas implementações de navegadores, tais como: *Internet Explorer, Microsoft Edge, Mozilla Firefox, Opera, Google Chrome*, entre outros. No entanto, o crescente número de implementações de navegadores e a constante evolução das características das tecnologias Web conduziram para diferenças em como os navegadores se comportam e processam as aplicações. Para superar este problema durante o processo de desenvolvimento de software, os desenvolvedores devem encontrar e corrigir os XBIs antes da implantação do sistema. Para detectar os XBIs, muitos destes desenvolvedores dependem dos testes manuais de cada página Web renderizada em várias configurações de ambientes (considerando sistema operacional e versões), independentemente dos esforços e custos que são necessários para realizar essas tarefas. **Objetivo**: Esta dissertação tem como objetivo propor uma abordagem de detecção automática de XBIs de *Leiaute*, baseada no uso de Aprendizado de Máquina, Segmentação da Árvore DOM e Comparação de Imagens. **Metodologia:** Para alcançar o objetivo desta pesquisa, o processo de Revisão Sistemática da Literatura (RSL) foi primeiramente executado para identificar o estado da arte neste tópico de pesquisa. Posteriormente, com base nos conhecimentos adquiridos, a abordagem proposta segmentou uma aplicação Web simples em múltiplos elementos DOM. A tarefa de detecção de XBI foi modelada como um problema de classificação de aprendizado de máquina (supervisionado) usando as seguintes propriedades para compor o conjunto de características: diferenças na posição, tamanho e comparação da imagem de cada elemento DOM de uma aplicação Web. **Resultados:** Validou-se a abordagem proposta em um experimento que investigou a eficácia do modelo de classificação. O experimento usou 64 aplicações Web composta de 5081 elementos DOM renderizados em três navegadores diferentes (*Google Chrome, Mozilla Firefox, Internet Explorer*). O experimento relatou resultados significativos de acurácia de acordo com a métrica *F-measure*, tendo atingido 0,91. **Conclusão:** Com os resultados obtidos, a abordagem proposta apresentou efetividade semelhante com o estado da arte.

Palavras-chave: Cross-Browser, XBI, Detecção de XBI.

ABSTRACT

PAES, Fagner Chrisitan. CROSS-BROWSER INCOMPATIBILITIES AUTOMATIC DETECTION USING MACHINE LEARNING AND SCREENSHOT SIMILARITY. 58 f. Dissertação de Mestrado – Programa de Pós-graduação em Informática, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2018.

Context: Cross-Browser Incompatibilities (XBIs) are compatibility issues that can be observed while rendering the same web application in different browsers. Users can interact with the Web through distinct browsers, such as: Internet Explorer, Microsoft Edge, Mozilla Firefox, Opera, Google Chrome, among others. However, the increasing number of browsers, and the constant evolution of web technologies led to differences in how browsers behave and render web applications. In order to overcome this issue during the software development process, web developers must detect and fix XBIs before deploying web applications. Many of these developers rely on manual tests of every web page rendered in several configuration environments (considering multiple platforms of operational systems and versions) to detect XBIs, regardless of the efforts and costs that are required to conduct these tasks. **Goal**: The goal of this research is to propose a approach of Layout XBIs automatic detection based on Machine Learning, Segmentation of the DOM Tree and Screenshot Comparison. **Method:** To reach the goal of this research, the process of Systematic Literature Review (SLR) was firstly executed identifing the current state of art for this research topic. Afterwards, based on the acquired knowledge, the proposed approach segmented a simple web application in multiple DOM elements. The task of XBI detection was modeled as a supervised learning classification problem using the following properties to compose the features set: differences in position, size and screenshot comparison of each DOM element of a web application. **Results:** The proposed approach was validated in an experiment that investigated the efficacy of the classification model. The experiment used 66 web application containing 5081 DOM elements rendered in three different browsers (Google Chrome, Mozilla Firefox, Internet Explorer). The experiment reported significant accuracy results according to F-measure metric having reached 0.91. **Conclusion:** The results validated the proposed approach with similar effectiveness as the state of art.

Keywords: Cross-Browser, XBI, XBI detection.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

XBIs	Cross-Browser Incompatibilities
SO	Sistema Operacional
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
W3C	World Wide Web Consortium
DOM	Document Object Model
RSL	Revisão Sistemática da Literatura
RS	Revisão Sistemática
QPs	Questões de Pesquisa
XML	Extensible Markup Language
API	Application Programming Interface
EMD	Earth Movers' Distance
ROI	Região de Interesse
FSCS	Fixed Size Candidate Set

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos as aplicações Web se tornaram mais populares, principalmente com o advento da Web 2.0 que proporcionou aos usuários maior interação na organização do conteúdo, pelo fato dos ambientes online estarem mais dinâmicos (WATANABE et al., 2015). Entretanto, os recentes avanços em termos de tecnologias, elevou a complexidade do desenvolvimento das aplicações Web.

Estas aplicações Web são construídas baseadas em uma tradicional arquitetura cliente-servidor, onde parte da aplicação contém componentes sendo executados no lado do servidor, através de uma variedade de linguagens de programação e a outra parte dos componentes sendo carregados e executados do lado do cliente, principalmente através de implementações em *JavaScript*[1] dentro dos navegadores Web. Atualmente, a fim de executar os componentes do lado do cliente, os usuários podem escolher uma ampla variedade de implementações de navegadores Web distintos, tais como: *Internet Explorer, Microsoft Edge, Mozilla Firefox, Opera, Google Chrome*, entre outros (CHOUDHARY et al., 2010b). Esta flexibilidade que caracteriza a portabilidade das aplicações Web, por outro lado, aumenta os custos e os esforços gastos no ciclo de desenvolvimento do software, uma vez que as aplicações Web devem suportar múltiplos navegadores.

Cada elemento de uma aplicação Web deveria ser corretamente renderizado e apresentar o mesmo comportamento, independentemente da implementação do navegador Web, versão ou Sistema Operacional (SO) utilizado pelo usuário (DALLMEIER et al., 2013). Em estudos recentes, as diferenças observadas na aparência ou comportamento de uma aplicação Web quando renderizada em diferentes implementações de navegadores têm sido referenciadas como *Cross-Browser Incompatibilities (XBIs)* (CHOUDHARY et al., 2010b, 2012). A Figura 1.1 ilustra um exemplo de XBI, note que cada navegador apresenta o mesmo elemento HTML da página Web de maneira diferente.

O teste manual para detectar os XBIs, exige que os desenvolvedores carreguem e inspecionem se as aplicações Web são apresentadas corretamente e se comportam de forma consistente nas diversas configurações de navegadores (SAAR et al., 2016). Essa tarefa é trabalhosa e consome muito tempo.

Atualmente, existem ferramentas comerciais que podem ser utilizadas para aliviar os custos desta atividade de teste manual e aleatório, tais como: *Microsoft Expression Web*[2]

[1] JavaScript: `https://www.w3.org/standards/webdesign/script.html`
[2] Microsoft Expression Web: `https://www.microsoft.com/expression/eng/`

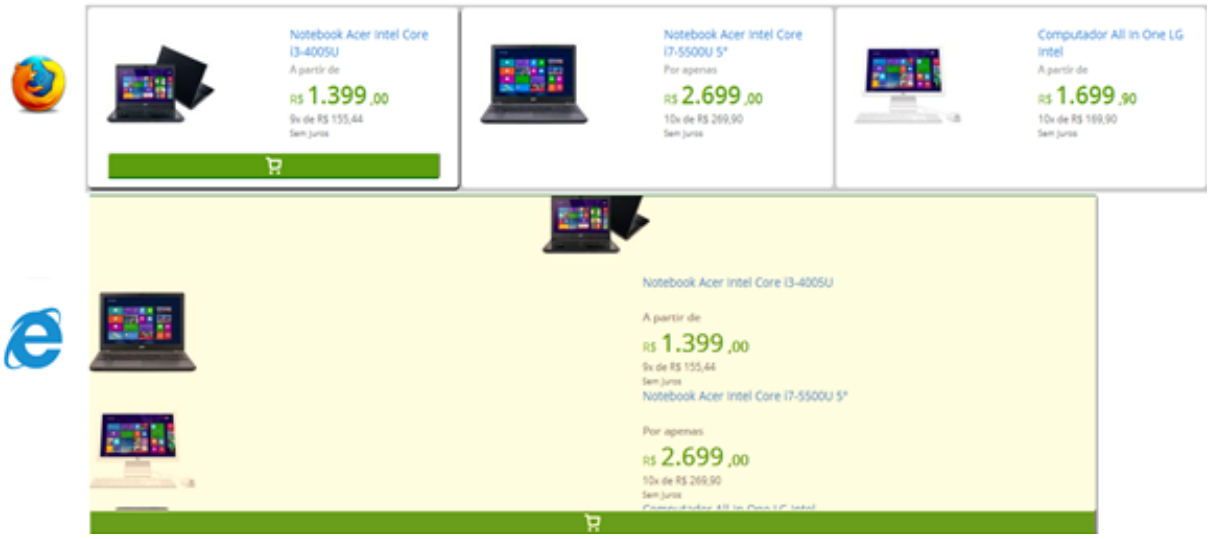


Figura 1.1: Exemplo de XBI entre os Navegadores.
Fonte: Autoria própria

e *Adobe Lab*[3]. Estas ferramentas geram automaticamente as imagens da captura de tela da aplicação Web renderizada em múltiplos navegadores. Entretanto, mesmo que estas ferramentas tornem o processo de identificação de XBIs mais fácil, desenvolvedores ainda dependem da inspeção manual das imagens para detectar os XBIs (CHOUDHARY et al., 2010a). Portanto, a detecção de XBIs exige esforços e custos elevados para o processo de desenvolvimento de software.

Nesta dissertação, o objetivo é melhorar a efetividade na detecção automática de XBIs baseada no uso de Aprendizado de Máquina, Segmentação da Árvore DOM e Comparação de Imagens. A abordagem proposta define a tarefa de detecção de XBI como um problema de classificação que modela o conjunto de características com base nas diferenças de posição, tamanho e imagem do elemento HTML da árvore DOM.

1.1 MOTIVAÇÃO

Uma aplicação Web é chamada de compatível com vários navegadores *(cross-browser)*, se for renderizada de maneira idêntica e funcionar corretamente nessas diferentes configurações. Na prática, manter a compatibilidade entre navegadores é uma tarefa desafiadora. Dado que, mesmo com as versões recentes de navegadores prontos para receber as novas tecnologias, tais como: HTML5, CSS3, Web 2.0, os desenvolvedores ainda enfrentam dificuldades em garantir o funcionamento das aplicações Web nessa variedade de navegadores, plataformas e versões presentes no mercado.

[3]Adobe Lab: `http://labs.adobe.com/`

Por isso, a cobertura de teste de *cross-browser* é muito importante, já que uma aplicação Web pode ser visualizada perfeitamente na versão mais recente de um determinado navegador, conter um visual e usabilidade incríveis. Todavia, quando disponibilizada na Internet, o usuário poderá acessá-la e visualizá-la com um aspecto completamente divergente em outro navegador, por exemplo. O fato da aplicação Web não funcionar de modo correto para o usuário, causa consequentemente uma falta de credibilidade, que pode resultar na perda financeira e de imagem para empresa proprietária da aplicação Web, visto que provavelmente este usuário não visitará mais a aplicação.

A detecção manual de incompatibilidades por meio de inspeção visual de páginas Web carregadas em múltiplos navegadores é uma tarefa trabalhosa e está sujeita a erros, pois os testadores tendem a perder algumas incompatibilidades devido ao cansaço (SEMENENKO, 2013). Por outro lado, a criação de técnicas de teste de *cross-browser* automatizadas significa reduzir a quantidade de esforço manual. Atualmente, as soluções existentes, tais como: WebDiff (CHOUDHARY et al., 2010a), CrossCheck (CHOUDHARY et al., 2012), X-Pert (CHOUDHARY et al., 2013), WebMate (DALLMEIER et al., 2012) e Broswerbite (SEMENENKO et al., 2013) possuem o propósito de detectar o problema de XBI. Entretanto, estas ferramentas sofrem de sensibilidade e produzem um montante de falsos positivos. Falsos positivos representam XBIs detectados automaticamente pelas abordagens, mas que não são observados na aplicação Web através de uma inspeção manual. A presença de falsos positivos diminui a precisão da abordagem.

O estudo de novas abordagens e ferramentas para redução de falsos positivos pode ser a chave para aumentar a efetividade na detecção automática de XBIs e garantir a qualidade das aplicações Web neste tópico de pesquisa.

1.2 OBJETIVOS

Os atuais avanços tecnológicos fizeram das aplicações Web mais dinâmicas. Consequentemente, trouxeram novos desafios para a engenharia Web, igualmente para atividade de desenvolvimento e testes de software. Por essas razões este projeto de pesquisa possui o seguinte objetivo:

- *Propor uma abordagem de detecção automática de XBIs de Leiaute baseada em Aprendizado de Máquina, Segmentação DOM e Comparação de Imagens.*

Primeiramente, para alcançar o objetivo nesta dissertação, foi conduzida uma Revisão

Sistemática da Literatura (RSL) com objetivo de identificar o estado da arte na detecção automática de XBI, compreender as abordagens existentes, destacar seus pontos fortes, pontos fracos. A fim de aprimorar o estado da arte, neste trabalho foi estabelecida a tarefa de detecção de XBI como um problema de classificação. Desta forma, foi avaliado o uso de aprendizado de máquina, segmentação da árvore DOM e a comparação de imagens. Posteriormente, um experimento foi realizado para investigar a eficácia da abordagem proposta. A metodologia do experimento foi separada em três atividades principais: aquisição de dados, classificação manual e avaliação do modelo.

O presente trabalho está diretamente relacionado a área de Engenharia Web, que dentre os seus objetivos encontram-se a facilidade da manutenibilidade e escalabilidade, minimização dos riscos e principalmente a melhoria e garantia da qualidade em aplicações Web (AHMAD et al., 2005). Por essas razões, as principais contribuições deste trabalho foram:

- *Apresenta uma Revisão Sistemática da Literatura com análise das diferentes abordagens utilizadas para detecção automática de XBIs, consideradas o estado da arte.*
- *Definição de abordagem de detecção automática de XBIs em aplicações Web com efetividade similar aos trabalhos relacionados.*

1.3 ESTRUTURA DO TRABALHO

Esta dissertação está estruturada da seguinte forma: o Capítulo 2 apresenta um processo de Revisão Sistemática da Literatura, bem como as diferentes abordagens para detecção automática de XBIs. No Capítulo 3 é descrito a abordagem proposta e as etapas realizadas para validação da eficácia do modelo. Por fim, no Capítulo 4 é realizada as considerações finais desta dissertação e apresentada as contribuições, limitações e trabalhos futuros.

2 REVISÃO DA LITERATURA

2.1 CONSIDERAÇÕES INICIAIS

Neste capítulo é apresentada uma visão geral dos assuntos que embasaram o tópico de pesquisa desta dissertação. Na Seção 2.2, uma introdução sobre o conceito de XBI é descrita. Na Seção 2.3 é apresentada uma Revisão Sistemática da Literatura com o objetivo de determinar o estado da arte na detecção automática de XBI, compreender técnicas distintas que foram utilizadas para identificar XBI e descrever a evolução das ferramentas desenvolvidas. Além disso, identificar potenciais lacunas nas pesquisas atuais, a fim de propor a nova abordagem. Na Seção 2.4 é explanada em detalhes as diferentes abordagens usadas para detecção automática de XBIs.

2.2 DETECÇÃO XBI

Uma página Web simples é construída basicamente com três principais tecnologias do lado do cliente: o HTML (*Hypertext Markup Language*)[1], que contém a estrutura de marcações de um texto, o CSS[2] (*Cascading Style Sheets*) utilizado para definir o estilo da página e o JS[3] (*JavaScript*) que lida com o comportamento dinâmico e a interatividade de uma página Web. Diante disso, o XBI representa características em cada uma dessas tecnologias (HTML, CSS e JS) que não são suportadas consistentemente nos navegadores. Logo, se uma aplicação Web fizer uso dessas características, então ela provavelmente apresentará XBIs.

A Figura 2.1 exemplifica uma página Web renderizada diferentemente em três navegadores distintos: *Google Chrome, Firefox* e *Internet Explorer*. As árvores DOM *(Document Object Model)*[4] em cada navegador foram renderizadas de forma distintas. Embora existam vários padrões, sugeridos pela W3C[5] que orientam o desenvolvimento para tornar a aplicação Web compatíveis aos navegadores, essa incompatibilidade comprova que ainda existem lacunas, e requisitos que podem ser implementados de formas diferentes pelos desenvolvedores.

De acordo com Choudhary et al. (2013), os XBIs podem apresentar desde problemas simples na interface do usuário até falhas críticas nas funcionalidades. Essas incompatibilidades

[1]HTML: `https://www.w3.org/standards/techs/html`

[2]CSS: `https://www.w3.org/standards/techs/css`

[3]JS: `https://www.w3.org/standards/webdesign/script.html`

[4]DOM: *Document Object Model* é o modelo de acesso de elementos HTML carregado em memória utilizando uma estrutura de dados de tipo árvore para qualquer página Web renderizada no navegador Web. Especificação disponível em: `http://www.w3.org/DOM/`.

[5]W3C: `http://www.w3.org`

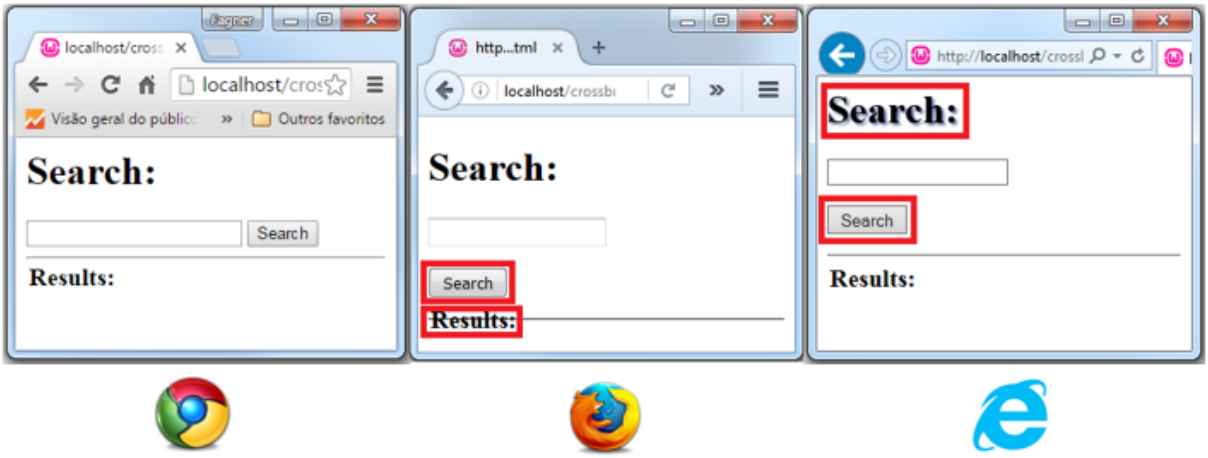


Figura 2.1: Uma simples página Web baseada em HTML/CSS/JS renderizada diferentemente no *Google Chrome, Firefox* e *Internet Explorer*.
Fonte: Autoria própria

podem ser classificadas em dois grupos:

- Inconsistência de Aparência: são causadas por diferenças em como o navegador inicialmente renderiza a página Web e são imediatamente visíveis pelo usuário. Esta inconsistência pode ser um posicionamento diferente, tamanho, visibilidade ou aparência em geral dos elementos de uma página Web; e
- Inconsistência de Funcionalidade: estão associadas ao comportamento da funcionalidade de uma aplicação Web e a maneira de como a página Web é executada em diferentes navegadores. Esta inconsistência pode limitar a maneira em que o usuário interage com a aplicação Web, dependendo do navegador que o usuário utiliza.

O estado da arte neste tópico de pesquisa, reportou o desenvolvimento de ferramentas para detecção automática de XBIs, tais como: WebDiff (CHOUDHARY et al., 2010a), CrossCheck (CHOUDHARY et al., 2012), X-Pert (CHOUDHARY et al., 2013), WebMate (DALLMEIER et al., 2012) e Broswerbite (SEMENENKO et al., 2013). Estas ferramentas propõem o uso da Análise da Estrutura DOM, Comparação de Imagens, Isomorfismo de Grafos, Aprendizado de Máquina e Comparação da Posição Relativa. Estas abordagens evoluíram ao longo dos últimos anos e foram implementadas separadamente em artigos, com o objetivo de aumentar sua eficácia ao detectar XBIs. Nas próximas seções deste capítulo, são apresentados em detalhes os resultados da Revisão Sistemática da Literatura neste tópico de pesquisa.

2.3 REVISÃO SISTEMÁTICA DA LITERATURA

Uma Revisão Sistemática da Literatura (RSL), muitas vezes conhecida simplesmente como Revisão Sistemática (RS), consiste de um estudo secundário do qual os objetivos são identificar, avaliar e interpretar todos os estudos relevantes para uma determinada questão de pesquisa, tópico ou fenômeno de interesse (KITCHENHAM, 2004). RSLs são focadas em reunir e sintetizar evidências de um tópico de pesquisa pré-estabelecido e como resultado determinar o estado da arte deste tópico.

No contexto da RSL, os "estudos primários" são estudos originais, cujos resultados contribuem para o propósito de uma determinada RSL (BUDGEN; BRERETON, 2006). Devido a isso, RSLs são geralmente referenciadas como "estudos secundários". Uma RSL deve ser conduzida seguindo algumas regras pré-definidas que são estabelecidas em um protocolo de pesquisa. Portanto, através de uma RSL, pesquisadores e profissionais têm por objetivo identificar todos os estudos relevantes em um tópico de pesquisa respeitando uma sequência organizada e criteriosa de passos. Isto torna o processo de RSL auditável e replicável, sendo uma das estratégias com menor risco para iniciar uma pesquisa. Também permite a avaliação do conhecimento científico sobre o tema de pesquisa. De acordo com Kitchenham (2004), as razões centrais para conduzir RSL são:

(i) Obter uma visão panorâmica sobre alguma tecnologia, por exemplo, resumir as evidências empíricas dos prós e contras do uso do método de especificação ágil;

(ii) Identificar potenciais lacunas em pesquisas atuais, a fim de sugerir novas investigações;

(iii) Fornecer estrutura e conhecimento confiável, a fim de posicionar adequadamente novas atividades de pesquisa.

Segundo Kitchenham (2004), um processo de RSL é conduzido em três etapas, conforme ilustrada na Figura 2.2.

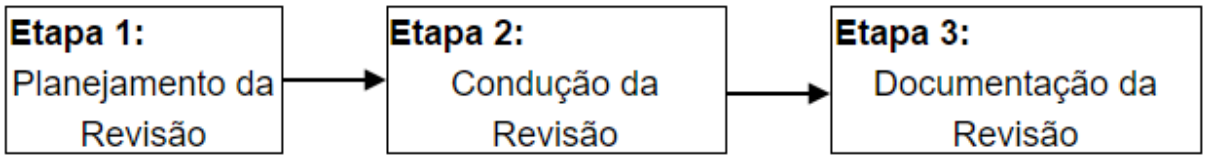


Figura 2.2: O processo da RSL em suas etapas.
Fonte: Adaptado de Brereton et al. (2007)

- Planejamento: Nesta etapa do processo, os pesquisadores devem conduzir uma atividade de planejamento, no qual alguns fatores chaves devem ser definidos: As questões de pesquisas e um protocolo de revisão detalhado. O protocolo da revisão deve estabelecer a origem da informação para os estudos, definir as *strings* de busca e os critérios de inclusão, exclusão e qualidade dos estudos.

- Condução da Revisão: Uma vez que o protocolo tenha sido criado, os pesquisadores estão autorizados a iniciar a revisão. Esta etapa do processo possui o objetivo de executar as seguintes tarefas: execução das buscas nas fontes determinadas na etapa de planejamento, seleção dos artigos baseados no critério de inclusão e exclusão de estudos, avaliação da qualidade dos estudos e execução da extração dos dados.

- Documentação da Revisão: Nesta etapa final aborda a interpretação e comunicação dos resultados da revisão sistemática. Disponibiliza os resultados para potenciais partes interessadas.

Neste estudo, foi conduzida uma RSL com o objetivo de identificar abordagens de detecção automáticas de XBIs. As próximas seções apresentam o planejamento da revisão sistemática, condução da pesquisa e resultados.

2.3.1 PLANEJAMENTO

2.3.1.1 OBJETIVOS

O objetivo da RSL é identificar as abordagens existentes na detecção automática de XBIs, destacando seus pontos fortes e fracos.

2.3.1.2 QUESTÃO DE PESQUISA

De acordo com Brereton et al. (2007), a definição das questões de pesquisa (QPs) é um dos pontos mais críticos em uma RSL.

As QPs são fundamentais para definir as *strings* de busca e para realizar as pesquisas em sistemas de busca de periódicos disponibilizados na Web. As QPs são extremamente importantes para definir o escopo da pesquisa, guiar o pesquisador em como coletar e analisar os estudos incluídos na RSL. As QPs são definidas no protocolo de avaliação e os pesquisadores podem refiná-las depois de conduzir algumas buscas pilotos para calibrar a *string* de busca.

Quanto à finalidade de identificar estratégias automatizadas de XBI, foram definidas duas QPs:

- *[QP1] Quais são as abordagens existentes para detecção automática de Incompatibilidades Cross-Browser (XBIs)?*
- *[QP2] Quais são as vantagens e desvantagens sobre a utilização das abordagens identificadas?*

2.3.1.3 MÉTODO DE BUSCA

O método adotado consiste da execução de uma *string* de busca em diferentes bases de dados disponibilizadas na *Web*. A *string* de busca genérica é apresentada abaixo:

- *Palavras-chave: ((“cross-browser” OR “cross browser”) AND (compatibility OR incompatibility OR testing OR test))*

A busca foi executada em três diferentes base de dados: ACM Digital Library [6], IEEE Xplore[7], Springerlink[8] e [9]. Foi necessário calibrar a *string* de busca para cada biblioteca de acordo com as regras particulares de cada mecanismo de busca.

2.3.1.4 CRITÉRIO DE INCLUSÃO E EXCLUSÃO

Com o objetivo de responder as QPs e com base no escopo da pesquisa, foram definidos os seguintes critérios de inclusão/exclusão:

- *Critério de Inclusão 1 (CI1): Todos os artigos que referenciam o problema de detecção de XBIs foram incluídos na RSL*;
- *Critério de Exclusão 1 (CE1): Todos os artigos que referenciam abordagens para prevenção de XBI durante a fase de codificação do processo de desenvolvimento de software foram excluídos;*
- *Critério de Exclusão 2 (CE2): Todos os artigos duplicados encontrados em mais de uma biblioteca digital foram excluídos; e*

[6] ACM Digital Library: `http://dl.acm.org/`
[7] IEEE Xplore:`http://ieeexplore.ieee.org`
[8] Springerlink: `http://link.springer.com/`
[9] ScienceDirect: `http://www.sciencedirect.com`

- *Critério de Exclusão 3 (CE3): Todos os artigos que não referenciam abordagens relacionadas a XBI foram excluídos.*

2.3.2 CONDUÇÃO DA REVISÃO

Com o objetivo de identificar estudos para compor a RSL, foi adaptada a *string* de busca genérica para carregar corretamente no motor de busca nas bibliotecas selecionadas. A busca foi realizada em duas etapas: A primeira busca foi realizada de 15/05/2015 até 12/09/2015 e depois disso, a busca foi atualizada de 15/08/2016 até 29/08/2016. A Tabela 2.1 apresenta o número de estudos retornados em cada biblioteca digital.

Tabela 2.1: Número de artigos idetificados na pesquisa por Bibliotecas Digitais.

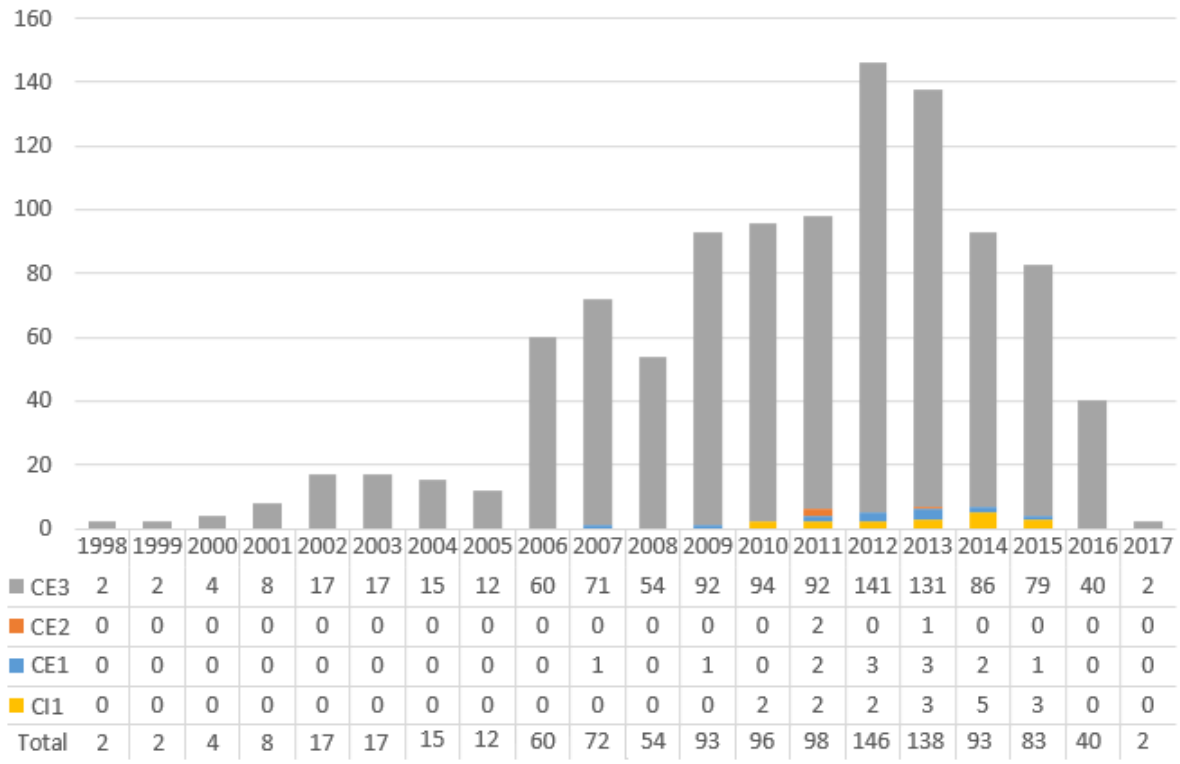
Biblioteca	Número de Artigos
ACM Portal	167
IEEEXplore	183
Springer	603
ScienceDirect	99
Total	1052

No processo de seleção foram aplicados os critérios de inclusão/exclusão/qualidade em todos os 1052 artigos identificados durante o processo de pesquisa. Estes critérios foram inicialmente aplicados no título e resumo dos artigos, resultando na identificação de 17 de 1052 artigos retornados na busca. A fim de torná-lo mais confiável, a aplicação dos critérios foram executados por um autor e revisados pelo outro autor.

2.3.3 RESULTADOS DA REVISÃO

A Figura 2.3 apresenta um gráfico de barra com o número total de artigos analisados na primeira etapa da RSL. Os artigos foram classificados inicialmente pelo ano de publicação. Os artigos foram publicados entre os anos de 1998 até 2017 (período de 19 anos). Além disso, houve um aumento no número de publicações depois de 2006, refletindo a popularidade das aplicações Web.

Depois de aplicado a classificação baseada nos CI/CEs, dos 1052 artigos retornados na busca 98% foram excluídos pelo critério CE3. Durante o processo de seleção, notou-se que vários artigos mencionaram o termo *cross-browser*, mas estes artigos não lidavam com a descrição de abordagens de detecção automática de XBIs. Na Figura 2.3 é possível observar o número de artigos que foram excluídos da RSL baseado nos critério de exclusão.



	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017
CE3	2	2	4	8	17	17	15	12	60	71	54	92	94	92	141	131	86	79	40	2
CE2	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	1	0	0	0	0
CE1	0	0	0	0	0	0	0	0	0	1	0	1	0	2	3	3	2	1	0	0
CI1	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	3	5	3	0	0
Total	2	2	4	8	17	17	15	12	60	72	54	93	96	98	146	138	93	83	40	2

Figura 2.3: Número de artigos encontrados entre 1998 até 2017 e classificados pelos critérios de inclusão e exclusão.

Fonte: Autoria própria

A Figura 2.3 também apresenta o total de artigos encontrados em relação ao ano de publicação. Ao analisar os 17 artigos incluídos na RSL, foi possível notar que os primeiros artigos sobre abordagens de detecção automática de XBIs foram publicados em 2010. Além disso, foi possível identificar pela RSL que a maioria dos artigos incluídos foram publicados em 2014.

De acordo com os números da primeira busca, 65% dos 17 artigos, foram publicados nos últimos três anos e 88% foram publicados nos últimos 5 anos. Isto demonstra que as abordagens XBI são constantemente investigadas pelos pesquisadores nos últimos 6 anos.

A Tabela 2.2 destaca os números de publicações de acordo com os pesquisadores e suas organizações. Neste contexto, a maioria das publicações foram realizadas pelos pesquisadores da Universidade *"Georgia Institute of Technology"*. Dentre os artigos selecionados, pode-se notar um total de 24 pesquisadores. O pesquisador S. Choudhary está envolvido em 35% do total de publicações. Além disso, o pesquisador R. Orso (29% do total de 17 publicações) e o pesquisador M. Prasad (23% do total de 17 publicações) possui o maior número de publicações depois do S. Choudhary. Dentre os artigos incluídos, os três pesquisadores publicaram 3 artigos juntos e 41% dos 17 artigos foram publicados individualmente ou como coautor com outros

pesquisadores.

Tabela 2.2: Pesquisadores e número de publicações.

Autores	Publicações	Organização/País
S. Choudhary	6	Georgia Institute of Technology, USA
A. Orso	5	Georgia Institute of Technology, USA
M. Prasad	4	Group Fujitsu Laboratories of America, USA
M. Burger	3	Saarland University, Germany
V. Dallmeier	3	Saarland University, Germany
A. Zeller	3	Saarland University, Germany
H. Zeng	3	School of Computer Engineering and Science, China
H. Versee	2	Georgia Institute of Technology
M. Dumas	2	University of Tartu, Estonia
T. Orth	2	Saarland University, Germany
T. Saar	2	Software Technology and Applications Competence Center, Estonia
N. Semenenko	2	University of Tartu, Estonia
M. Kaljude	1	Software Technology and Applications Competence Center, Estonia
A. Mesbah	1	University of British Columbia Vancouver, Canada
M. Mirold	1	Testfabrik, German
B. Pohl	1	Testfabrik, German
E. Selay	1	University of Wollongong, Australia
Z. Zhou	1	University of Wollongong, Australia
L. Sanchez	1	Fundação Educacional Inaciana, Brazil
P. Aquino	1	Instituto de Pesquisas Tecnológicas, Brazil
J. Zou	1	University of Petroleum, China
X. Li	1	School of Computer Engineering and Science, China
S. Xu	1	School of Computer Engineering and Science, China
H. Shi	1	School of Computer Engineering and Science, China

A próxima seção apresenta uma análise detalhada das abordagens identificadas na RSL.

2.4 ABORDAGENS DE DETECÇÃO AUTOMÁTICA DE XBI

As abordagens de detecção automática de XBIs consistem em soluções tecnológicas distintas que foram usadas para identificar incompatibilidades em aplicações Web quando renderizadas em diferentes navegadores. Esta seção apresenta os resultados da RSL conduzida para extrair as diferentes abordagens usadas na academia para detecção automática de XBIs.

Todos os estudos na RSL reportaram XBIs na aplicação Web executadas do lado do cliente, através dos navegadores que não suportam características *cross-browser*.

Vale a pena ressaltar que dos 17 artigos selecionados na RSL, foram identificadas 7 ferramentas para detecção de XBIs. Além disso, existem múltiplos artigos que reportaram experimentos diferentes e conduziram avaliações na mesma ferramenta. Existem também, estudos que desenvolveram soluções tecnológicas para identificação de XBI, mas não implementaram ou nomearam uma ferramenta. As ferramentas identificadas no estudo são apresentadas na Tabela 2.3.

Cada estudo selecionado nesta RSL propõe abordagens diferentes, os quais podem concentrar se em um grupo específico de XBIs. As abordagens tecnológicas identificadas nesta revisão foram: **Análise da Estrutura DOM (AED)**, **Comparação de Captura da Tela (CCT)**, **Isomorfismo de Grafos (IG)**, **Aprendizado de Máquina (AM)**, **Posição Relativa (PR)**, **Testes Aleatórios Adaptativos (TAA)** e **Análise Estática (AE)**. A Tabela 2.3 lista os estudos selecionados no processo de RSL, apresenta quais abordagens foram implementadas por cada estudo e a ferramenta correspondente. A Tabela 2.3 também apresenta o número de estudos que exploraram cada abordagem tecnológica para detecção de XBIs. A Figura 2.4 ilustra o número de artigos que utilizam cada abordagem tecnológica para detecção de XBIs.

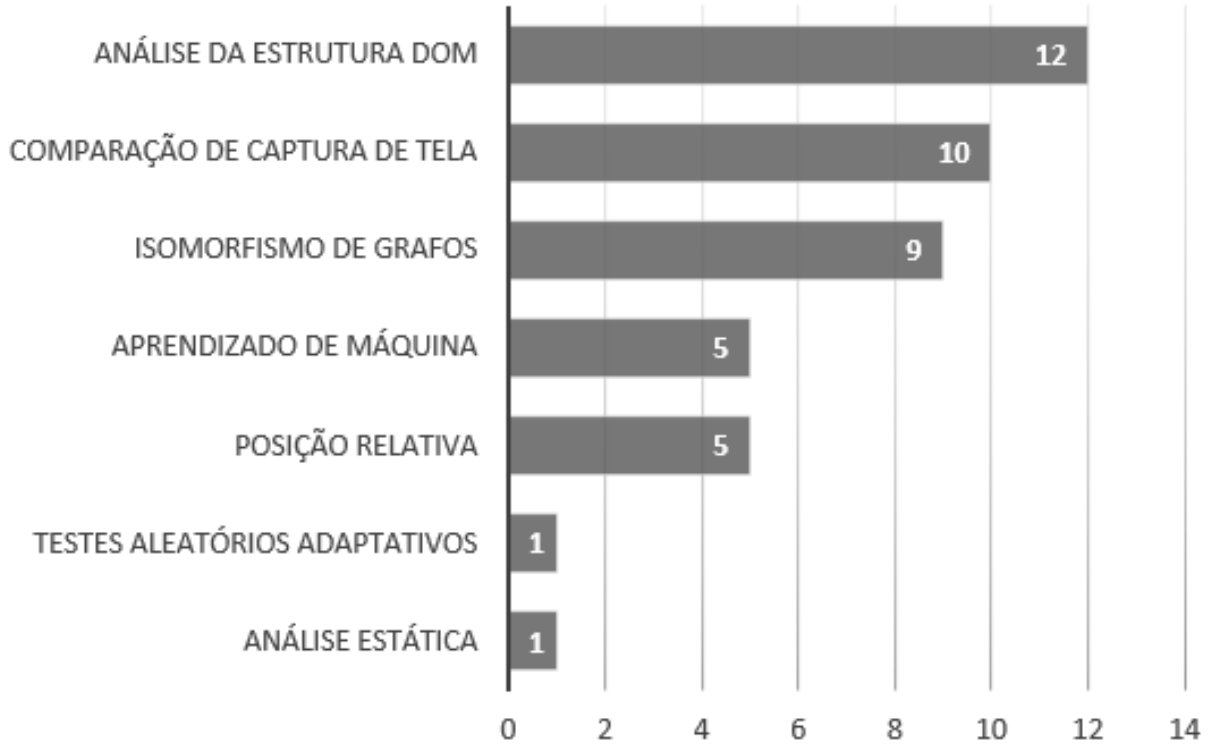


Figura 2.4: Número de artigos que implementaram cada abordagem tecnológica para detectar XBIs.

Fonte: Autoria própria

A próxima subseção apresenta as diferentes abordagens usadas para identificação de XBIs. As abordagens foram ordenadas de forma cronológica, ou seja, quando a solução tec-

Tabela 2.3: Artigos selecionados na RSL, as ferramentas que foram implementadas em cada artigo e as abordagens tecnológicas que foram usadas para detectar XBIs.

	Artigo	AED	CCT	IG	AM	PR	TAA	AE	Ferramenta
1	WebDiff: Automated identification of cross-browser issues in Web applications	X	X	-	-	-	-	-	WebDiff
2	A cross-browser Web application testing tool	X	X	-	-	-	-	-	WebDiff
3	Detecting cross-browser issues in Web applications	X	X	-	-	-	-	-	WebDiff
4	Automated cross-browser compatibility testing	X	-	X	-	-	-	-	CrossT
5	CrossCheck: Combining crawling and differencing to better detect cross-browser incompatibilities in Web applications	X	X	X	X	-	-	-	CrossCheck
6	WebMate: A tool for testing Web 2.0, applications	X	-	X	-	X	-	-	WebMate
7	X-Pert: Accurate identification of cross-browser issues in Web applications	X	X	X	X	X	-	-	X-Pert
8	Browserbite: Accurate cross-browser testing, via machine learning over image features	-	X	-	X	-	-	-	Browserbite
9	WebMate: Web application test generation in the real world	X	-	X	-	X	-	-	WebMate
10	Modeling Web application for cross-browser compatibility testing	X	-	X	-	-	-	-	-
11	X-Pert: A Web application testing tool for, cross-browser inconsistency detection	X	X	X	X	X	-	-	X-Pert
12	Adaptive random testing for comparison, in regression Web testing	-	X	-	-	-	X	-	-
13	WebMate generating test cases, for Web 2.0	X	-	X	-	X	-	-	WebMate
14	Cross-browser testing in Browserbite	-	X	-	X	-	-	-	Browserbite
15	Automatic deformations detection in internet interfaces: ADDII	-	X	-	-	-	-	-	ADDI
16	Static Analysis Technique of Cross-Browser Compatibility Detecting	-	-	-	-	-	-	X	-
17	Cross-browser compatibility testing based on model comparisson	X	-	X	-	-	-	-	-
-	Total	12	10	9	5	5	1	1	7

nológica foi proposta pela primeira vez nos estudos.

2.4.1 ANÁLISE DA ESTRUTURA DOM

O DOM define uma estrutura lógica do documento HTML e XML permitindo aos programas e *scripts* acessarem e atualizarem o conteúdo dinamicamente. O DOM representa os documentos HTML e XML como uma estrutura de dados em árvore e os navegadores frequentemente fornecem uma API (*Application Program Interface*) para ser usada pela aplicação Web.

A maioria dos estudos selecionados no processo de RSL usaram a estrutura DOM da página Web para detectar XBIs. Estas abordagens consistem em comparar atributos e elementos de representações DOM renderizados em diferentes navegadores para detectar XBIs.

Inicialmente, múltiplas representações DOM de uma página Web renderizada em diferentes navegadores foram capturadas. Então, estas representações DOM foram comparadas por uma outra similar, a fim de identificar os atributos e elementos que não apresentam o mesmo valor (CHOUDHARY et al., 2010b, 2010a; CHOUDHARY, 2011; MESBAH; PRASAD, 2011; CHOUDHARY et al., 2012; DALLMEIER et al., 2012; CHOUDHARY et al., 2013; DALLMEIER et al., 2013, 2014; LI; ZENG, 2014; CHOUDHARY et al., 2014; SHI; ZENG, 2015). Nestas abordagens, se o mesmo elemento em diferentes representações DOM apresentou valores de atributos incompatíveis, então este elemento era marcado com um XBI e reportado para os desenvolvedores. Nesta etapa, a maioria dos estudos utilizaram na verificação um grupo selecionado de atributos.

Choudhary et al. (2010b) publicaram o primeiro artigo que utilizou esta abordagem. A mesma abordagem foi futuramente desenvolvida em outros artigos, tais como (CHOUDHARY et al., 2012, 2013; MESBAH; PRASAD, 2011). As aplicações Web ao longo do tempo evoluíram, logo observou-se que apenas comparar os atributos na estrutura DOM renderizados em diferentes navegadores, levaram a um alto número de falsos positivos. Assim, os estudos que utilizam esta técnica não consideram apenas comparação das representações DOM, mas utilizam outras estratégias, como a normalização dos valores de atributos, aprendizado de máquinas, entre outras técnicas que são descritas nas próximas subseções desta RSL.

Ainda considerando análise da estrutura DOM, os estudos (CHOUDHARY et al., 2010b, 2012, 2013) apresentaram implementações detalhadas de como combinar elementos de diferentes representações DOM. Representações DOM renderizadas em múltiplos navegadores podem apresentar diferenças estruturais que dificultam a tarefa de comparar os atributos

e elementos que podem ser identificado como falha dependendo do navegador em que ele foi renderizado. A partir destes estudos apresentados, uma técnica baseada no uso de atributos, tais como: *tagName*, *id*, *xpath* e um *hash mapping* do conteúdo interno (*innerHTML*) foram utilizados para combinar elementos similares de diferentes representações DOM.

A RSL encontrou 12 artigos e 5 ferramentas que utilizaram a abordagem *Análise da Estrutura DOM*.

2.4.2 COMPARAÇÃO DE CAPTURA DA TELA

As informações da estrutura DOM são utilizadas para auxiliar o processo de identificação de XBI nas páginas Web. Entretanto, a estrutura DOM não apresenta informações sobre a exata aparência de cada elemento na tela (CHOUDHARY et al., 2010b). Assim, no que diz respeito à identificação de XBIs de **Inconsistência de Aparência**, muitos estudos relataram o uso de técnicas de comparação de imagens capturadas a partir da aplicação Web (CHOUDHARY et al., 2010b, 2010a; CHOUDHARY, 2011; CHOUDHARY et al., 2012, 2013; SEMENENKO et al., 2013; CHOUDHARY et al., 2014; SELAY et al., 2014; SAAR et al., 2014; SANCHEZ; JR., 2015).

O primeiro método para comparação de imagens é uma comparação direta dos histogramas. No entanto, utilizar uma comparação rígida dos histogramas sem recursos globais (como a simetria, curtose e números de picos) para identificar XBIs pode encontrar muitos falsos positivos (SAAR et al., 2014). Neste contexto, as 6 principais técnicas de comparação de imagem utilizadas nos estudos foram: *Earth Movers' Distance* (EMD) (CHOUDHARY et al., 2010b, 2010a; CHOUDHARY, 2011), $\chi 2$ *Distance* (CHOUDHARY et al., 2012, 2013, 2014), *Image Segmentation* e *Correlation-based Image Comparison* (SEMENENKO et al., 2013; SAAR et al., 2014), *pHash* e *Perceptual Image Diff* (SANCHEZ; JR., 2015).

O EMD é baseado em uma solução para os problemas de transporte de otimização linear. Como uma técnica de comparação de imagem, o EMD é uma metodologia mais robusta que a técnica de comparação de histograma (RUBNER et al., 2000). O EMD é uma medida de distância que pode ser aplicada para distribuição de texturas e cores.

Choudhary et al. (2010b, 2010a), Choudhary (2011), relataram o uso de EMD para comparar individualmente imagens da tela capturadas em diferentes navegadores para cada elemento da estrutura DOM da página Web. Nestes estudos, Choudhary (2011) não usaram um único limiar para determinar se existe ou não incompatibilidade entre duas imagens do elemento. O limiar (*threshold*) foi configurado considerando o tamanho e densidade da cor de

cada imagem.

A técnica de comparação de imagem $\chi 2$ *Distance* é adaptada da estatística $\chi 2$ que representa um teste de hipótese que é usado para comparar a dispersão entre duas variáveis nominais, ou seja, avalia a correlação existente entre as variáveis qualitativas. O princípio básico do teste de hipótese $\chi 2$ é comparar proporção de taxas com o objetivo de identificar diferenças na frequência em um evento que foi detectado em dois grupos. Assim, dois grupos são classificados como similar se as diferenças na frequência observada e na frequência esperada para múltiplas categorias são muito pequenas ou perto de zero. Como uma técnica de processamento de imagem, o método $\chi 2$ *Distance* foi utilizado com sucesso para classificar objetos, textura e imagens duplicadas (PELE; WERMAN, 2010).

Os estudos (CHOUDHARY et al., 2012, 2013, 2014) utilizaram o método de $\chi 2$ *Distance* para comparar imagens dos elementos da página Web. Inicialmente, em (CHOUDHARY et al., 2013, 2014), foi utilizada para comparar todos os elementos da página Web. Já outro estudo, (CHOUDHARY et al., 2012), o método $\chi 2$ *Distance* foi usado para comparar somente os nós folhas da estrutura da árvore DOM da página Web. Esta estratégia de comparação somente da imagem do nó folha foi usada para reduzir falsos positivos na avaliação de XBIs.

Choudhary et al. (2013, 2014) também relataram que o método $\chi 2$ *Distance* é menos custoso computacionalmente e apresenta melhor precisão quando comparado com o método EMD. Os estudos (SEMENENKO et al., 2013; SAAR et al., 2014) relataram a segmentação de imagens renderizadas em diferentes navegadores em pequenos retângulos identificados como Região de Interesse (ROI) baseado na mudança de cor e bordas observadas na imagem. Sendo assim, a comparação da imagem foi feita entre o mesmo ROI capturado em diferentes navegadores. Estes estudos mencionam o uso de uma técnica de comparação de imagem baseada em correlação.

Sanchez e Aquino Jr. utilizaram duas abordagens diferentes para comparar imagens da página Web: o *pHash* [10] e *Perceptual Image Diff* [11] (SANCHEZ; JR., 2015). Ambos os algoritmos foram utilizados de maneira complementares, ao passo que *pHash* foi utilizado para reportar se as imagens são semelhantes e o *Perceptual Image Diff* foi aplicado para apresentar a quantidade de *pixels* que são diferentes. O algoritmo *pHash* é uma implementação do conceito *Perceptual Hash* que é definido como uma impressão digital *(fingerprint hash)* de um arquivo multimídia derivado de várias recursos do seu conteúdo.

O *Perceptual Image Diff*, por outro lado, é um algoritmo que identifica cada *pixel*

[10] pHash: `http://pastebin.com/Pj9d8jt5`
[11] Perceptual Image Diff: `http://pdiff.sourceforge.net`

em duas imagens diferentes. Primeiro, o *pHash* foi aplicado em cada imagem capturada nos diferentes navegadores para gerar o código *Hash*. Além disso, o código *Hash* de cada imagem foi comparado utilizando o algoritmo de distância *Hamming*. O resultado do algoritmo de distância *Hamming* foi empregado junto com o método *Perceptual Image Diff* para avaliar se um XBI foi identificado ou não.

A RSL encontrou 10 artigos e 5 ferramentas que utilizaram a abordagem *Comparação da Captura de Tela*.

2.4.3 ISOMORFISMO EM GRAFOS

Os estudos (MESBAH; PRASAD, 2011; CHOUDHARY et al., 2012; DALLMEIER et al., 2012, 2014; CHOUDHARY et al., 2013; DALLMEIER et al., 2013; CHOUDHARY et al., 2014; LI; ZENG, 2014; SHI; ZENG, 2015) relataram o uso de uma abordagem separada que focou a identificação de XBIs em **inconsistências funcionais**. Esta abordagem consiste em simular cenário de interação do usuário com a aplicação, principalmente eventos de cliques em elementos da página Web, para disparar mudanças na representação da árvore DOM da aplicação Web em um navegador específico. Em seguida, a mesma aplicação Web estaria sujeita a mesma simulação de cenário da interação do usuário em outro navegador, e assim, compara as representações DOM obtidas entre o primeiro e segundo navegador para verificar se são iguais. Para realizar esta avaliação, os estudos mapearam as representações DOM e simulações do evento clique dentro de uma representação gráfica e, em seguida, utilizaram o algoritmo Grafo de Isomorfismo para identificar funcionalidades que não estavam disponíveis dependendo do navegador que o usuário estava utilizando.

O primeiro estudo que empregou esta abordagem foi (MESBAH; PRASAD, 2011). Este estudo estendeu uma ferramenta *crawler* para aplicações Ajax, chamada de Crawljax (MESBAH; DEURSEN, 2009; MESBAH et al., 2012). Estudos relacionados ao Crawljax propõem o uso de simulações de eventos em elementos da página Web, por exemplo, eventos de cliques para capturar mudanças na estrutura DOM.

Aplicações Web em Ajax apresentam uma estrutura DOM dinâmica que pode ser alterada durante interações do usuário. A fim de ser capaz de mapear estas interações, o *crawler* é colocado para gerar eventos nos elementos da página Web, disparar mudanças e acessar potenciais informações que não foram disponíveis na estrutura DOM da página Web quando carregada inicialmente no navegador.

Mesbah e Prasad utilizaram Crawljax para disparar mudanças na estrutura DOM. Uma

vez que os eventos de cliques foram enviados para a página Web e mapeados os dados gerados em uma máquina de estado finito, os estados denotaram as múltiplas gerações da estrutura DOM e a ações dos usuários representaram as transições (os cliques que disparavam cada mudança da estrutura DOM). A representação de nível superior desta máquina de estado finito é um conceito gráfico nomeada como Modelo de Navegação. Sua abordagem seria mapear vários modelos de navegação, um para cada navegador. Em seguida, esse paradigma de navegação renderizado em diferentes navegadores seria comparado utilizando um processo de verificação de equivalência, especificamente a execução de um algoritmo de Isomorfismo de Grafo.

Em relação ao procedimento de verificação de equivalência, o problema do Grafo de Isomorfismo consiste em identificar uma estrutura completa de um grafo, dentro de outro grafo (BUCHANAN et al., 1984). Mesbah e Prasad executou um algoritmo de Isomorfismo de Grafos e todas incompatibilidades identificadas por este algoritmo foram relatadas como XBIs de **inconsistências funcionais**, representando funcionalidades de cliques que foram observadas em um navegador, mas não no outro (MESBAH; PRASAD, 2011).

Depois dos estudos do Mesbah e Prasad, outras abordagens de detecção XBI implementaram estratégias similares, tais como (CHOUDHARY et al., 2012; DALLMEIER et al., 2012, 2014; CHOUDHARY et al., 2013; DALLMEIER et al., 2013; CHOUDHARY et al., 2014).

A RSL encontrou 9 artigos e 4 ferramentas que utilizaram a abordagem *Isomorfismo de Grafos*.

2.4.4 APRENDIZADO DE MÁQUINA

Aprendizado de Máquina é uma área da Inteligência Artificial, onde o objetivo é desenvolver técnicas computacionais que são capazes de adquirir automaticamente o conhecimento sobre como executar uma tarefa específica, sem ser explicitamente programada para fazê-la. Weiss e Kulikowski definem aprendizado de máquina como um programa de computador que toma decisões baseado em experiências adquiridas e acumuladas através de soluções bem sucedidas de problemas anteriores (WEISS; KULIKOWSKI, 1991).

Um problema contínuo enfrentado pelos estudos pesquisados sobre soluções de detecção XBI foi o número alto de falsos positivos identificados por suas técnicas. Em relação a isso, um grupo de pesquisadores investigaram o uso das técnicas de aprendizado de máquina para reduzir o número de falsos positivos em suas soluções de detecção XBI.

As técnicas de aprendizado de máquina são divididas em três principais categorias:

aprendizado supervisionado, não supervisionado e por reforço. Todos os estudos de detecção XBI que foram selecionados nesta RSL utilizaram técnicas de aprendizado supervisionado. Em aprendizado supervisionado, conceitos são induzidos a partir de exemplos que são pré-classificados.

As abordagens de detecção XBIs que utilizaram aprendizado de máquina, usaram exemplos de códigos em HTML/CSS/JS de páginas Web que continham XBIs e exemplos de códigos que não continham. Estes exemplos foram aplicados como dados de aprendizagem para as técnicas de aprendizado de máquina distintas. O uso de aprendizado de máquina entre os estudos de detecção XBI, alteraram em quais características de uma aplicação Web e quais técnicas são utilizadas na fase de aprendizagem.

Os primeiros estudos que utilizaram técnicas de aprendizado para detecção de XBI foram (CHOUDHARY et al., 2012, 2013, 2014). Estes estudos relataram o uso de árvore de decisão, uma técnica de aprendizado de máquina. A árvore de decisão é uma simples representação de um classificador utilizado por muitos sistemas de aprendizado de máquina, tal como C4.5 (QUINLAN, 1993). Estes artigos utilizaram árvore de decisão para classificar cada elemento da estrutura DOM renderizado em diferentes navegadores que apresentou um XBI ou não. Nos artigos, a árvore de decisão foi treinada considerando as seguintes características:

1. **Relação da diferença de tamanho**: diferenças no tamanho dos mesmos elementos renderizados em diferentes navegadores;
2. **Deslocamento**: a distância euclidiana entre a posição dos mesmos elementos renderizados em diferentes navegadores;
3. **Área**: a área mais pequena dos mesmos elementos renderizados em diferentes navegadores;
4. **Diferenças de Texto no Nó Folha**: um valor Booleano identifica se existe diferença entre os textos dos mesmos elementos renderizados em diferentes navegadores; e
5. **Distância da Imagem**: a comparação das imagens do mesmo elemento, renderizado em diferentes navegadores através da técnica $\chi 2$ *Distance*.

Além disso, Semenenko et al. (2013) e Saar et al. (2014) relataram o uso de árvore de decisão e redes neurais para detectar XBIs em aplicações Web. Redes neurais representam um modelo matemático inspirado na estrutura neural que adquire conhecimento através da experiência. A característica essencial da rede neural está relacionada a capacidade em aprender

e melhorar como ela executa uma tarefa específica através de treinamento contínuo (HEATON, 2008). Semenenko et al. (2013) e Saar et al. (2014) realizaram o experimento em um grupo de páginas Web para comparar ambas abordagens de aprendizado de máquina. Semenenko et al. (2013) e Saar et al. (2014) utilizaram região de interesse para identificar XBIs, diferentemente de outras abordagens que utilizaram elementos DOM da página Web. Portanto, a abordagem de rede neural destes autores foi executada para cada região de interesse renderizado em dois navegadores, através das seguintes 17 características:

- **10 histogramas** de imagens das regiões de interesse de uma página Web renderizadas em dois navegadores diferentes;
- **Comparação da imagem** baseada em correlação entre cada captura de tela na mesma região de interesse renderizada em dois navegadores;
- **Posição vertical** e **horizontal** da região de interesse renderizada em dois navegadores;
- **Tamanho vertical** e **horizontal** da região de interesse renderizada em dois navegadores;
- **Índice de Configuração** identificando os navegadores que foram usados para renderizar a região de interesse; e
- **Métrica de densidade de incompatibilidade**, que é calculada como um coeficiente do número de regiões de interesses que apresentam diferenças na abordagem de comparação de imagem baseada na correlação e o número total de região de interesse da página Web.

As técnicas de aprendizado de máquinas (árvore de decisão e redes neurais) foram usadas para reduzir o número de falsos positivos na identificação de XBI. Choudhary et al. (2013, 2014) relataram que o uso de árvore de decisão melhorou os resultados fornecidos pela sua abordagem de detecção de XBI.

Nos estudos Semenenko et al. (2013) e Saar et al. (2014), redes neurais apresentou melhor desempenho em comparação a árvore de decisão. No entanto, é difícil comparar (CHOUDHARY et al., 2012, 2013, 2014) com (SEMENENKO et al., 2013; SAAR et al., 2014), uma vez que os estudos utilizaram um conjunto diferente de características e treinamento.

A RSL encontrou 5 artigos e 3 ferramentas que utilizaram a abordagem *Aprendizado de Máquina*.

2.4.5 POSIÇÃO RELATIVA

Algumas técnicas anteriormente mencionadas utilizaram a posição dos elementos e região de interesse para detectar XBIs. Em relação a isto, Dallmeier et al. (2012, 2013, 2014) propuseram que a comparação entre os elementos considerasse também a posição relativa.

As páginas Web e os elementos HTML apresentam uma estrutura hierárquica e renderização de leiaute, portanto, se um elemento pai está desalinhado em uma página Web, os elementos filhos também apresentarão o mesmo posicionamento absoluto desalinhado. Desta maneira, as abordagens que utilizam posição absoluta para identificação de XBIs podem reportar que o pai e todos os elementos filhos representam XBIs. Entretanto, somente o elemento pai estava realmente desalinhado.

Choudhary et al. (2013, 2014) também exploraram esta estratégia de avaliação da posição relativa de leiaute. A fim de utilizar posição relativa na abordagem de detecção de XBI, Choudhary et al. (2013) representaram o posicionamento dos elementos na página Web em um grafo, chamado *Alignment Graph*. O *Alignment Graph* representa as relações entre os nós pai, filhos e irmãos como elementos no grafo. Já a sua posição relativa em relação a um outro elemento é representada como transições entre cada elemento no grafo. Em seguida, dois grafos de alinhamento extraídos de dois navegadores diferentes podem ser comparados por equivalência e qualquer diferença observada entre os grafos é relatado um XBI.

A RSL encontrou 5 artigos e 2 ferramentas que utilizaram a abordagem *Posição Relativa*.

2.4.6 TESTES ALEATÓRIOS ADAPTATIVOS

Testes Aleatórios Adaptativos - TAA é uma extensão da técnica *Testes Aleatórios*. O TAA analisa o comportamento dos resultados dos testes observados em software e gera dados de entrada que são mais propensos a produzir falhas no software (CHEN et al., 2010). Em geral, casos de teste semelhantes tendem identificar falhas semelhantes, consequentemente TAA é uma técnica que identifica casos de teste que melhoram a cobertura de teste de um domínio de entrada do software.

Selay et al. (2014) utilizaram TAA para comparar imagens por equivalência, considerando uma abordagem de comparação de imagem para identificar XBI. Selay et al. (2014) implementaram o algoritmo *Fixed Size Candidate Set* (FSCS) para escolha de casos de teste aleatórios na abordagem TAA. Este algoritmo faz uso de uma métrica de similaridade do caso de teste para garantir geração de diversos casos de teste que exploram diferentes partes do

domínio de entrada do software (CHEN et al., 2010).

Os artigos anteriores tinham o objetivo de melhorar a eficácia das abordagens em detecção de XBIs, no entanto, diferentemente dos demais autores Selay et al. (2014) executaram um experimento com o objetivo de melhorar o desempenho em termos do tempo requerido para identificar diferenças enquanto compara duas imagens distintas.

A RSL encontrou 1 artigo e nenhuma ferramenta que utilizou a abordagem *TAA*.

2.4.7 ANÁLISE ESTÁTICA

Análise Estática é uma técnica que é aplicada diretamente em artefatos do software, portanto, não executa o software durante a análise. Xu e Zeng (2015) construíram um conjunto de dados com características incompatíveis do HTML5 que consiste de elementos e atributos que não são suportados pelos três navegadores mais populares e propôs um algoritmo para detectar características incompatíveis em uma aplicação Web alvo baseado na análise estática do código HTML. A entrada do algoritmo consiste do código da aplicação Web alvo e as características HTML5 não suportadas pelo navegador em teste. A saída do algoritmo é uma lista de XBIs da aplicação Web alvo.

A técnica utiliza expressão regular para detectar elementos HTML na aplicação Web e identificar diferentes características HTML.

A RSL encontrou 1 artigo e nenhuma ferramenta que utilizou a abordagem *Análise Estática*.

2.4.8 LIMITAÇÕES DA REVISÃO SISTEMÁTICA

O processo de RSL foi conduzido a fim de identificar todas abordagens de detecção automática de XBI que foram utilizadas no estado da arte. Entretanto, em relação a validade desta RSL, podem haver estudos que reportam o uso de técnicas de detecção XBI mas que não foram incluídos na RSL. A fim de reduzir os riscos associados a esta ameaça específica, o processo de RSL foi conduzido por dois pesquisadores, considerando que todos os estudos e a aplicabilidade dos critérios de inclusão e exclusão foram discutidos e revisados entre eles. Estudos sistemáticos da literatura podem ser conduzidos seguindo duas técnicas de pesquisa: bola de neve e base de dados (BADAMPUDI et al., 2015).

No processo da RSL foi realizado pesquisas por artigos relacionados a técnica de detecção XBI nas bases de dados: ACM, IEEEXplore, Springerlink e ScienceDirect. Desta

forma, os resultados desta RSL poderiam ser melhorados se outras bases de dados fossem incluídas e a técnica bola de neve também fosse aplicada.

Contudo, a inclusão de outros estudos na RSL implicaria na adição de outras abordagens relacionadas com uma das abordagens já descritas nesta RSL ou, no máximo, adicionaria uma nova categoria de abordagem para os resultados desta RSL. Além disso, não invalidaria quaisquer resultados relatados anteriormente. Portanto, essa RSL pode ser atualizada com outras fontes e o uso da técnica de bola de neve em trabalhos futuros.

2.5 CONSIDERAÇÕES FINAIS

Neste capítulo, foi conduzido um Processo de Revisão Sistemática da Literatura com o objetivo de identificar as diferentes abordagens que foram utilizadas para detecção automática de XBIs, considerando o estado da arte. Foi observado que muitos estudos utilizaram as técnicas de Análise da Estrutura DOM e Comparação de Captura da Tela para detecção automática de XBIs. Além disso, com o objetivo de reduzir o número de falsos positivos, foi reportado na maioria dos estudos envolvidos, o uso de algoritmos de Aprendizado de Máquina, Isomorfismo de Grafos, Posição Relativa e Testes Aleatórios Adaptativos.

3 ABORDAGEM DE DETECÇÃO AUTOMÁTICA DE XBI DE LEIAUTE

3.1 CONSIDERAÇÕES INICIAIS

Neste capítulo é apresentada uma abordagem que possibilita detectar automaticamente XBIs de Leiaute, baseada no uso de aprendizado de máquina e segmentação da árvore DOM que compõe uma aplicação Web. Na seção 3.2, a tarefa de detecção de XBI é modelada como um problema de classificação de aprendizado supervisionado. Pretende-se, por meio da abordagem proposta, contribuir com a diminuição de falsos positivos existentes nas técnicas consideradas no estado da arte. Na seção 3.3, é avaliado a eficácia do modelo proposto para o problema de classificação. As limitações e discussões dos resultados são apresentadas na seção 3.4 e 3.5 respectivamente.

3.2 ABORDAGEM PROPOSTA

Esta pesquisa denota uma abordagem de detecção de XBI de Leiaute que segmenta uma simples aplicação Web através dos elementos HTML existentes na árvore DOM. Portanto, a identificação de XBI em aplicação Web foi dividida em múltiplas tarefas de detecção de XBI em elementos DOM. O XBI de Leiaute foi classificado em duas categorias (Figura: 3.1):

XBI Interno: são inconsistências relacionadas com a formatação do texto, plano de fundo e efeitos visuais do elemento DOM. Para avaliar o elemento DOM nesta categoria, são capturados em cada navegador o ***Screenshot*** que representa o elemento DOM.

XBI Externo: são inconsistências que indicam diferenças na posição (*top, left*) e tamanho (*height, width, area*) do elemento DOM. Já para avaliar cada elemento DOM nesta categoria, as seguintes propriedades são capturadas nos elementos em cada navegador considerado na abordagem:

Posição: dois valores numéricos (*top, left*) que representam a posição do elemento em relação ao canto superior esquerdo da janela do navegador.

Tamanho: três valores numéricos (*height, width, area*) que representam o tamanho, a largura e a área de um elemento DOM.

A abordagem proposta também determina um procedimento específico para capturar estas propriedades em uma aplicação Web. Com a finalidade de armazenar essas características

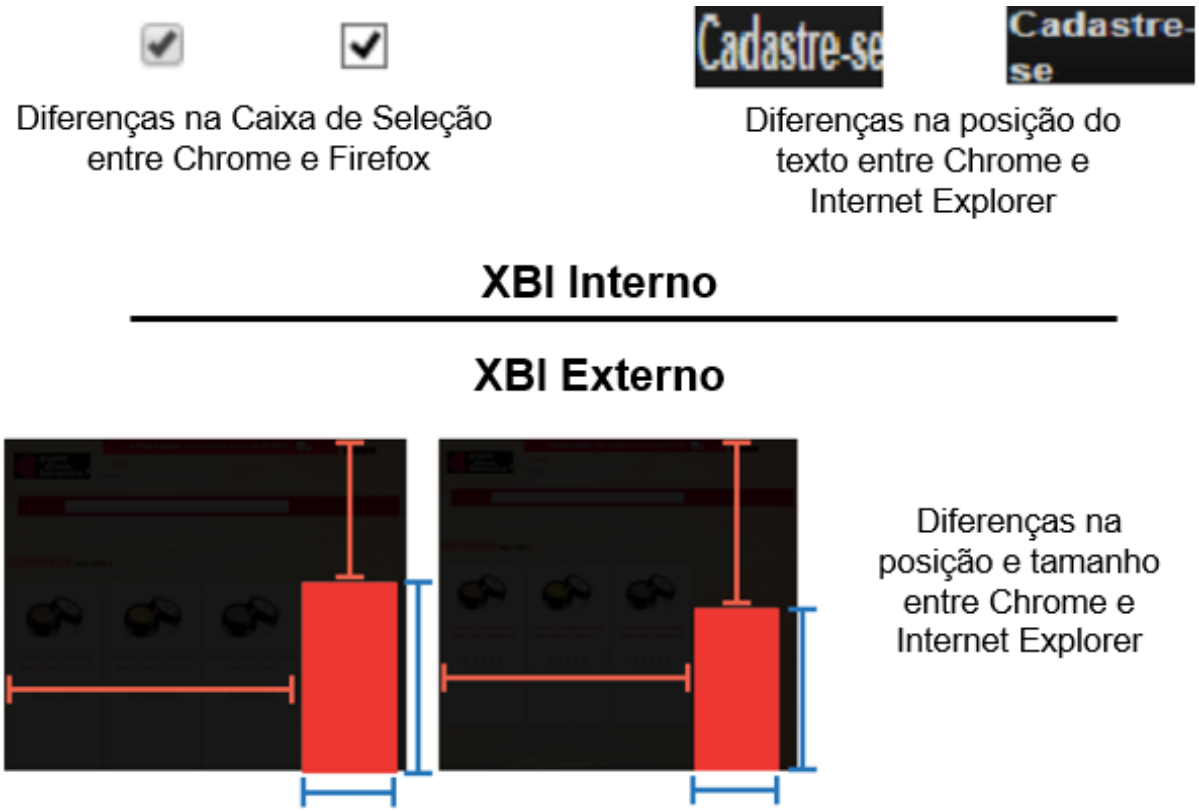


Figura 3.1: Exemplos de XBI interno e XBI externo.
Fonte: Extraído de Paes e Watanabe (2018)

do elemento, o processo é iniciado a partir dos nós folhas da árvore DOM. Após as propriedades (posição, tamanho e *screenshot*) do elemento DOM serem capturadas, o procedimento oculta o elemento da árvore DOM. Essas características adicionais capturadas, foram conduzidas para que tais inconsistências associadas a um elemento, não se propagassem ou impactassem a análise de seu nó pai na estrutura DOM, possivelmente melhorando a eficácia da abordagem proposta. A Figura 3.2 ilustra o processo de segmentar e ocultar os elementos previamente inspecionados na árvore DOM.

Depois de capturá-las em cada navegador, a abordagem definiu um navegador referência e comparou suas propriedades de elementos DOM com as mesmas coletadas em outros navegadores (os navegadores alvos). A Figura 3.3 ilustra como a comparação entre as propriedades dos elementos DOM do navegador referência (*Browser 0*) e do navegador alvo (*Browser 1*) foram usadas para modelar as características em um problema de classificação. O problema de classificação elaborado nesta pesquisa foi modelado com as seguintes características:

Diferenças de Posição: são as diferenças detectadas nas comparações das propriedades *(top, left)* que representam a posição do mesmo elemento DOM capturado no navegador referência e no navegador alvo. A posição dos elementos DOM foi usada como uma carac-

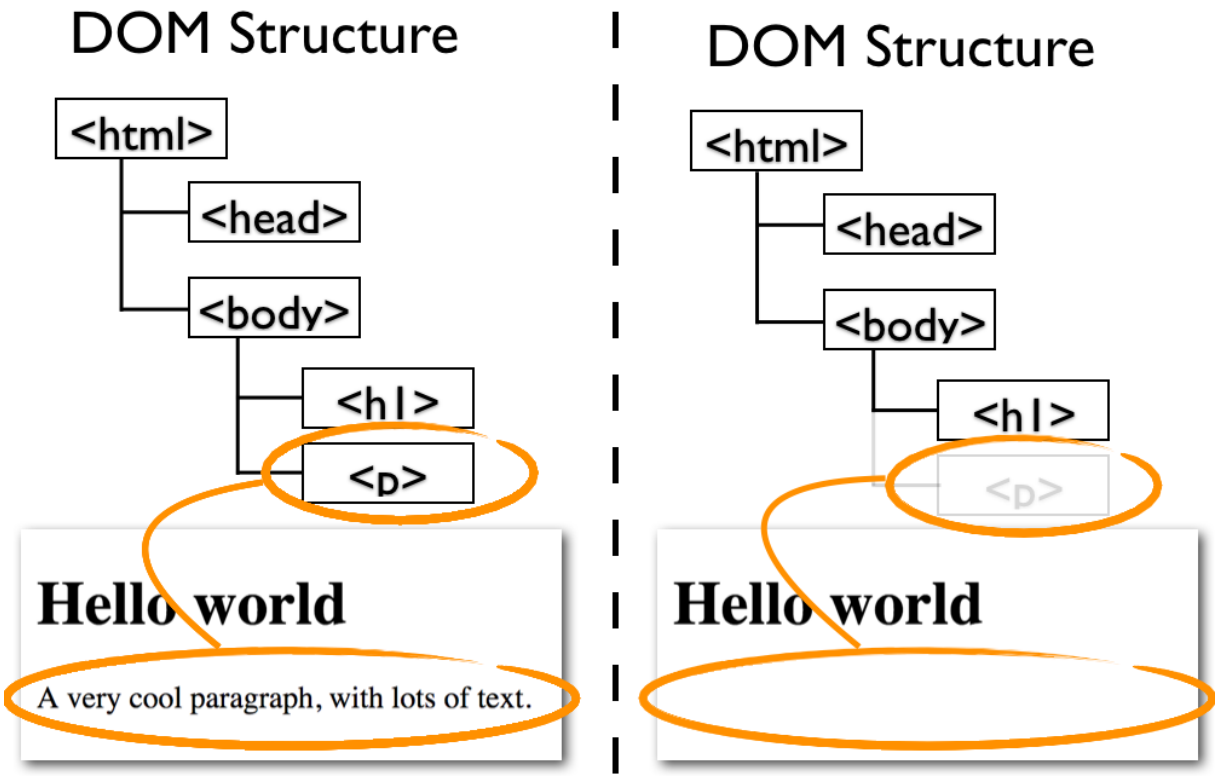


Figura 3.2: Processo de segmentar e ocultar os elementos na árvore DOM.
Fonte: Extraído de Paes e Watanabe (2018)

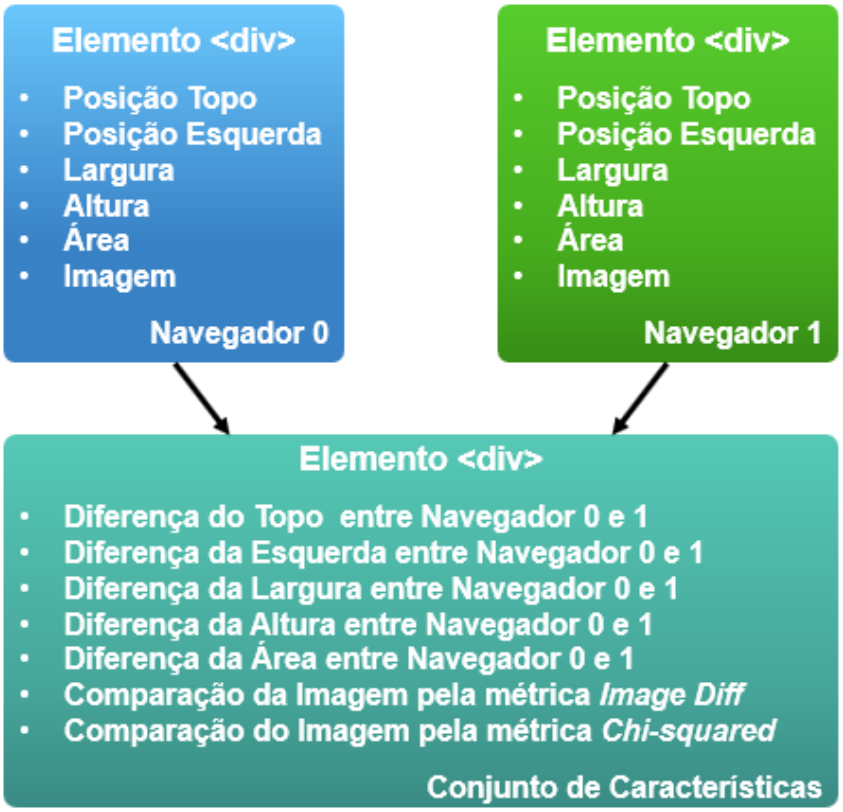


Figura 3.3: Características para detecção de XBI.
Fonte: Extraído de Paes e Watanabe (2018)

terística para identificar XBIs externos associados ao deslocamento do elemento DOM, dependendo do mecanismo de renderização de um navegador específico.

Diferenças de Tamanho: são as diferenças de altura, largura e área do mesmo elemento DOM capturado no navegador referência e no navegador alvo. As diferenças de tamanho entre os elementos DOM também foram usadas como características para identificar XBI externo associado as diferenças na dimensão do elemento DOM, dependendo do mecanismo de renderização de um navegador específico.

Comparação de *Screenshots*: diferenças entre as imagens capturadas do elemento DOM no navegador referência e no navegador alvo. As diferenças das imagens foram calculadas em conformidade com as duas métricas:

- **Image diff**: a diferença ou distância média da cor para cada pixel das imagens.
- **χ2 distance**: distância entre os histogramas das imagens capturadas.

As métricas de comparação de *screenshots* foram utilizadas como características para identificar XBIs internos, associados as alterações no texto, plano de fundo, efeitos visuais e outras características de um elemento DOM. Ambas as métricas foram usadas de forma complementar para aumentar a eficácia na detecção de XBIs internos.

Quando os resultados das comparações das propriedades apresentam valores próximos de zero, significam diferenças mínimas entre os elementos, não representando um XBI. Porém, diferenças maiores possivelmente indicam XBI. A próxima seção apresenta a avaliação do problema de classificação proposto.

3.3 AVALIAÇÃO

Com o objetivo de validar a abordagem proposta, foi conduzido um experimento para investigar a eficácia do modelo de classificação na detecção de XBIs de Leiaute. Para auxiliar no procedimento de avaliação da proposta deste trabalho foram estabelecidas as seguintes hipóteses a serem testadas:

H0: *O modelo de classificação proposto não é capaz de detectar automaticamente XBIs de Leiaute em aplicações Web.*

H1: *O modelo de classificação proposto é capaz de detectar automaticamente XBIs de Leiaute em aplicações Web.*

Se os algoritmos de classificação reportarem valores altos na métrica *F-measure*, então os resultados do experimento apresentarão evidências que suportam a hipótese **H1** e a hipótese **H0** é rejeitada. Portanto, pelos resultados do experimento, o modelo de classificação proposto é capaz de detectar automaticamente XBIs de Leiaute em aplicações Web. Entretanto, se os algoritmos de classificação reportarem valores baixos, então os resultados do experimento apresentarão evidências que rejeitam a hipótese **H1**.

3.3.1 METODOLOGIA

A metodologia do experimento foi separada em três etapas sequenciais, conforme ilustrada na Figura 3.4 e detalhada a seguir:

1. **Aquisição de Dados** *(Data Acquisition)*: segmenta as aplicações Web através da estrutura da árvore DOM; coleta as propriedades e o *screenshot* de cada elemento para gerar o conjunto de características; e remove a visibilidade dos elementos DOM, de acordo com a descrição na Seção 3.2. As informações extraídas são utilizadas na construção do conjunto de dados do experimento. O conjunto contém as propriedades e a imagem de cada elemento da árvore DOM de três navegadores (*Mozilla Firefox, Internet Explorer* e *Google Chrome*).

2. **Classificação Manual** *(Manual Classification)*: nesta etapa, as propriedades e a imagem de cada elemento DOM extraídas dos navegadores (referência e o alvo) foram comparadas manualmente e classificadas como um XBI de Leiaute ou não.

3. **Avaliação de Modelo** *(Model Evaluation)*: o algoritmo de árvore de decisão foi configurado com a finalidade de construir três modelos de árvore de decisão (*unpruned, winnowed* e *winnowed* com 5, 10 e 20 iterações *boost*) para treinamento e avaliação do modelo através do conjunto de dados. A fim de reduzir a chance de viés foi utilizado o procedimento *Cross-Validation 10-Folds* que divide os dados em 10 subconjuntos e repete os 10 experimentos de treino e avaliação, cada vez mantendo um conjunto diferente para avaliação.

As próximas subseções apresentam as três etapas da metodologia utilizada no experimento.

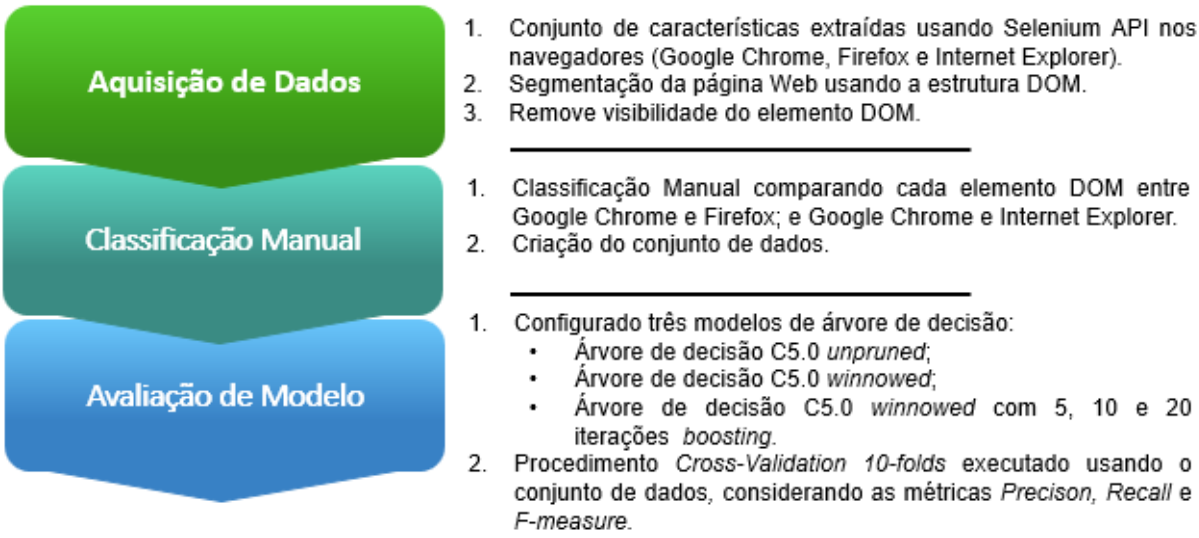


Figura 3.4: Metodologia Experimental.
Fonte: Extraído de Paes e Watanabe (2018)

3.3.1.1 AQUISIÇÃO DE DADOS

Para realizar a atividade de aquisição de dados, um script *crawler* foi implementada. A aplicação *crawler* foi construída usando o *Selenium Webdriver API*[1] na linguagem de programação *JAVA*, para abrir sistemicamente a aplicação Web em múltiplos navegadores e coletar as propriedades dos elementos DOM para cada navegador configurado. Durante o experimento, o *crawler* foi configurado para utilizar os seguintes navegadores: *Mozilla Firefox, Internet Explorer* e *Google Chrome*. Todos os navegadores foram executados no sistema operacional Windows 8.

O *crawler* foi executado em um conjunto de 64 aplicações Web, desenvolvidas por alunos de cursos de desenvolvimento Web com pouca experiência em programação. Consequentemente, os desenvolvedores não apresentavam conhecimentos prévios ou experiência com problemas de compatibilidade *cross-browser* em aplicações Web. Este fator pode ter impactado o número e variedade de XBIs que foram observados no experimento. Além disso, as 64 aplicações Web replicaram o Leiaute de sites brasileiros de alto tráfico, tais como, e-commerce, portais de notícias, sites institucionais, e entre outros exemplos. A complexidade da estrutura DOM das páginas Web também variou de acordo com o domínio do Leiaute da aplicação Web replicada. As aplicações Web do tipo e-commerce, por exemplo, apresentaram até 280 elementos DOM, enquanto alguns sites institucionais apresentaram até 40 elementos DOM. As aplicações Web foram implementadas apenas com *HTML* e *CSS*, sem uso de *JavaScript*, representando versões mais simples de aplicações Web do mundo real.

[1] Selenium Webdriver API: `http://www.seleniumhq.org/`

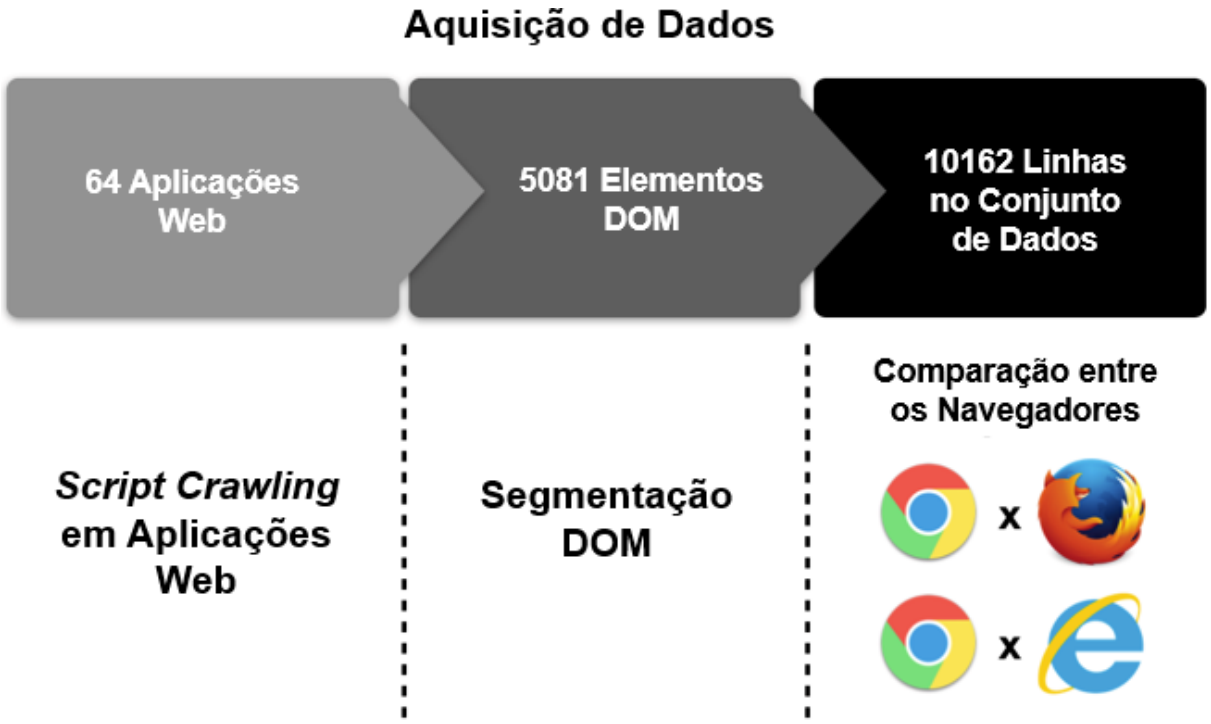


Figura 3.5: Definição do Conjunto de Dados Completo.
Fonte: Extraído de Paes e Watanabe (2018)

O *crawler* identificou o total de 5.081 elementos DOM para cada navegador nas 64 aplicações Web analisadas. Neste projeto de pesquisa o navegador *Google Chrome* foi definido como navegador referência, porque atualmente é o navegador Web mais popular do mundo.[2] Logo o conjunto de características foi comparado através das propriedades dos 5.081 elementos DOM do *Google Chrome* com os 5.081 elementos DOM do *Mozilla Firefox*. Também foi utilizado o mesmo critério de comparação entre os elementos DOM do *Google Chrome* com os elementos DOM do *Internet Explorer*. Portanto, depois de executado o passo anterior, foi construído um conjunto de características de 10.162 comparações entre o navegador referência (*Google Chrome*) e os navegadores alvos (*Firefox, Internet Explorer*). Os passos executados são ilustrados na Figura 3.5.

3.3.1.2 CLASSIFICAÇÃO MANUAL

Após adquirir o conjunto de características com o total de 10.162 comparações entre o navegador referência e os navegadores alvos, cada comparação foi sujeita a uma classificação manual de XBI. Essa classificação foi executada considerando a aparência dos elementos DOM nos múltiplos navegadores e foi desempenhada por dois engenheiros de software Web experi-

[2]Estatísticas dos Navegadores: `https://www.w3schools.com/browsers/browsers_chrome.asp` e `http://gs.statcounter.com/browser-market-share/desktop/`

entes. Para reduzir a chance de viés na avaliação manual, esse processo foi revisado por um terceiro especialista em *Design Web*. Apresentando um resultado de 2.097 XBIs: 1.233 XBIs identificados quando comparado *Google Chrome* com *Internet Explorer* e 864 XBIs identificados quando comparado *Google Chrome* com *Firefox*. Os resultados da classificação manual, juntamente com as características extraídas para cada uma das 10.162 comparações formou o conjunto de dados usado no experimento.

3.3.1.3 AVALIAÇÃO DE MODELO

Neste experimento, foi utilizada a árvore de decisão C5.0[3] que é um algoritmo de classificação de aprendizado de máquina supervisionado. Os classificadores de árvore de decisão são as representações de classificadores mais populares (ROKACH; MAIMON, 2005), principalmente devido à sua capacidade de capturar preditores não lineares com base nas características de entrada (QUINLAN, 1986). Os *scripts* executados no experimento e todas as análises estatísticas relatadas na seção de resultados foram implementados na linguagem R. Além disso, o algoritmo foi configurado para construir as seguintes árvores de decisão:

- árvore de decisão *unpruned*;
- árvore de decisão *winnowed*;
- árvores de decisão *winnowed* com 5, 10 e 20 iterações *boost*.

Com objetivo de reduzir as chances de viés no procedimento *Cross-Validation 10-Folds*, um processo de geração de subconjuntos *(folds)* foi conduzido para separar completamente as aplicações Web. De modo que, as propriedades dos elementos DOM de uma aplicação Web específica não fossem divididas em subconjuntos diferentes. A mesma aplicação Web tende apresentar XBIs semelhantes, portanto, agrupando várias comparações de elemento DOM das mesmas aplicações Web no mesmo subconjunto pode reduzir as chances de viés no experimento.

No experimento, avaliou-se a acurácia dos algoritmos de aprendizado de máquina (supervisionado) na detecção de XBIs de Leiaute quando modelado com o conjunto de características proposto. A eficácia foi medida com as seguintes métricas (POWERS, 2011):

Precison : representa a proporção de XBIs relevantes em relação a todos os XBIs identificados (verdadeiro positivos e falso positivos) pela abordagem de classificação.

[3] C5.0: `https://cran.r-project.org/web/packages/C50/index.html`

Recall : representa a proporção de XBIs relevantes em relação ao conjunto completo de XBIs existentes (verdadeiro positivos e falso negativos) na aplicação Web.

F-measure : é a média harmônica ponderada, calculada através das métricas *Precision* e *Recall*.

A próxima subseção apresenta os resultados do experimento.

3.3.2 RESULTADOS

Os resultados obtidos nas métricas *Precision*, *Recall* e *F-measure* no procedimento *Cross-Validation 10-Folds* para os classificadores de árvore de decisão são apresentados na Tabela 3.1. As seguintes configurações foram utilizadas:

C5.0 noGlobalPrunning: um classificador de árvore de decisão *unpruned*.

C5.0 iterations=1: um classificador de árvore de decisão usando estratégia de seleção de características *winnowing*.

C5.0 iterations=5: um classificador de árvore de decisão usando estratégia de seleção de características e estratégia de 5 iterações *boosting*.

C5.0 iterations=10: um classificador de árvore de decisão usando estratégia de seleção de características e estratégia de 10 iterações *boosting*.

C5.0 iterations=20: um classificador de árvore de decisão usando estratégia de seleção de características e estratégia de 20 iterações *boosting*.

Tabela 3.1: Os resultados de *Precision, Recall e F-measure* obtidos no procedimento *Cross-Validation 10 Fold* variando os parâmetros de configuração do algoritmo C5.0

	iterations=20	iterations=10	iterations=5	iterations=1	noGlobalPrunning
Precision	0.941317203	0.934556616	0.888059110	0.898643857	0.891978181
Recall	0.891532594	0.892183852	0.900899898	0.860489654	0.857940453
F-measure	0.913558096	0.910760524	0.892811042	0.875416976	0.868761112

Em relação a métrica *Precision*, os valores variaram entre 0,89 e 0,95. Valores mais elevados implicam em um número reduzido de falsos positivos. Isto significa que houveram poucos XBIs identificados pela abordagem de classificação que não eram verdadeiros. A abordagem de classificação que apresentou o melhor resultado foi a **C5.0 iterations=20**. Este classificador de árvore de decisão corresponde a configuração de modelo mais complexo que foi

testada, usando uma estratégia de seleção de características, modelos de árvore de decisão múltiplas e um sistema de votação entre eles.

Na métrica *Recall*, os resultados oscilaram entre 0,85 e 0,91, o que implica na redução de falsos negativos em valores mais elevados. Neste caso, a abordagem de classificação conseguiu identificar uma maior proporção de XBIs existentes na aplicação Web. O classificador que apresentou o melhor resultado da métrica *Recall* foi o **C5.0 iterations=5**.

Por fim, a métrica *F-measure*, que representa a combinação das duas métricas anteriores (*Precision, Recall*), os valores variaram entre 0,86 e 0,92. A abordagem de classificação que apresentou o melhor resultado para esta métrica foi o **C5.0 iterations=20**. Em relação aos resultados, a abordagem de classificação que apresentou os melhores resultados gerais foi a configuração **C5.0 iterations=20**.

Os resultados da métrica *F-measure* gerados pelo procedimento *Cross-Validation 10-Folds* foram sujeitos a uma análise estatística. A Figura 3.6 ilustra os diagramas de densidade dos resultados do *F-measure 10-Folds*. Os testes de normalidade de *Shapiro Wilk* foram executados nas métricas *F-measure* relatadas por cada abordagem de classificação. Foram identificadas duas delas como distribuições *non-normal* (em um intervalo de confiança 0,95):

- **C5.0 noGlobalPrunning** com W=0.84212 *p-value*=0.02256;
- **C5.0 iterations=1** com W=0.85219 e *p-value*=0.03044.

O teste de *Friedman* foi executado considerando a distribuição *non-normal* observada nos grupos de dados.

Apesar da pequena amostra (composta de 10 subconjuntos), os resultados do teste de *Friedman* indicaram que a métrica *F-measure* era diferente entre os classificadores, com *Friedman chi-squared*=27,958, *df*=4 e *p-value*=$1.272 * 10^{-5}$ em um intervalo de confiança de 0,95. A análise *Post-hoc* usando o teste de comparação múltipla *(Pairwise Nemenyi)* revelou diferenças significativas entre:

- **C5.0 iterations=20** e **C5.0 iterations=1**, com p-value=0.0062;
- **C5.0 iterations=20** e **C5.0 noGlobalPruning**, com p-value=0.0022;
- **C5.0 iterations=10** e **C5.0 iterations=1**, com p-value=0.0037;
- **C5.0 iterations=10** e **C5.0 noGlobalPruning**, com p-value=0.0013.

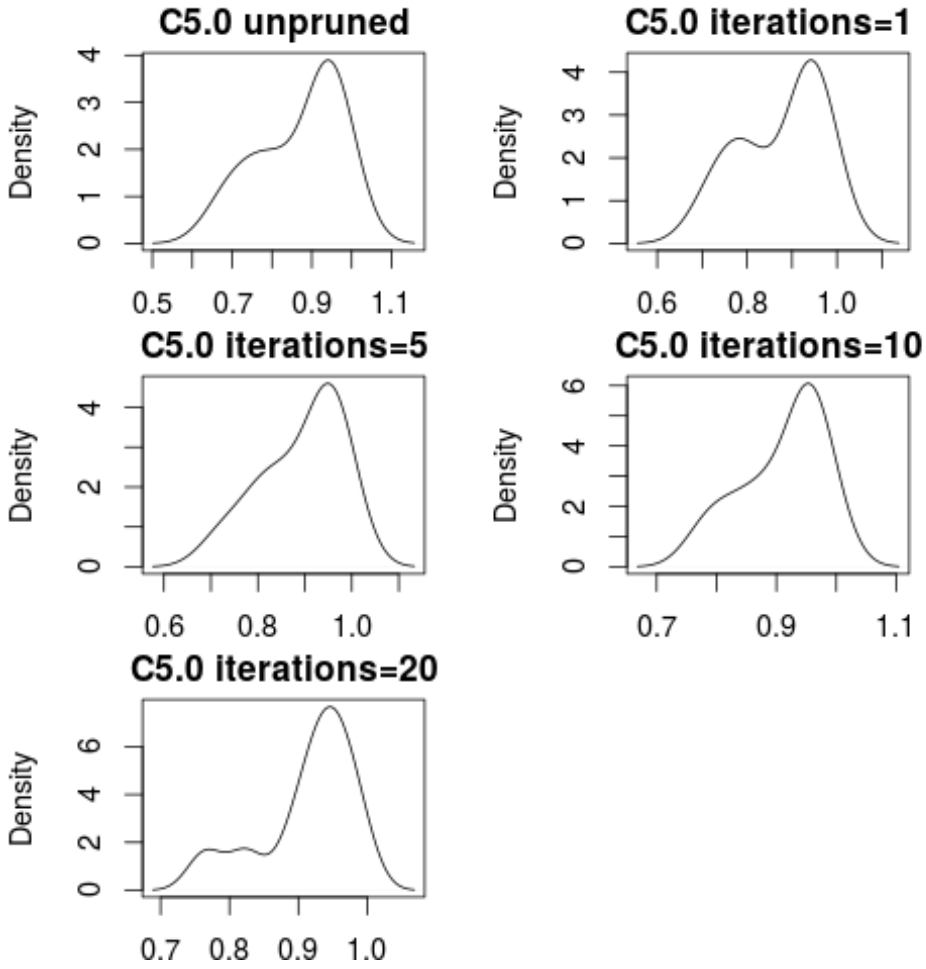


Figura 3.6: Distribuição de densidade *F-measure* para todos os experimentos utilizando algoritmo C5.0

Fonte: Extraído de Paes e Watanabe (2018)

Todas as comparações dos resultados *p-value* que são reportados pelo teste de comparação múltipla *Pairwise Nemenyi* estão retratadas na Tabela 3.2.

Tabela 3.2: *P-values* de um teste *Nemenyi post-hoc* no *F-measure* entre todas as configuração do algoritmo C5.0

	iterations=20	iterations=10	iterations=5	iterations=1	noGlobalPrunning
iterations=20	-	-	-	-	-
iterations=10	0.9999	-	-	-	-
iterations=5	0.7502	0.6639	-	-	-
iterations=1	0.0062	0.0037	0.1826	-	-
noGlobalPruning	0.0022	0.0013	0.0963	0.9986	-

Em relação a hipótese do experimento, os valores elevados de *Precision*, *Recall* e *F-measure* reportados, representam evidências que suportam **H1**. Assim, o modelo de classificação proposto foi capaz de detectar XBIs de Leiaute no conjunto de dados, observado no procedimento de *Cross-Validation*. Além disso, o classificador que melhor funcionou em geral foi

o **C5.0 iterations=20**, apresentando resultados de *F-measure* significativamente diferentes em comparação com os classificadores mais simplistas: **C5.0 iterations=1** e **C5.0 noGlobalPruning**. No entanto, não houve significância estatística quando comparado o **C5.0 iterations=20** com **C5.0 iterations=10** ou **C5.0 iterations=5**.

3.4 LIMITAÇÕES

Durante o experimento não foi conduzido uma comparação direta da eficácia da abordagem proposta em relação aos trabalhos relacionados. Este experimento alcançou o valor de 0,91 para a métrica *F-measure*, isso significa um nível de acurácia semelhante ao relatado pela ferramenta Browserbite em seu próprio experimento. Apesar disso, os resultados de precisão obtidos em ambos os estudos não são categóricos, devido as diferenças nas configurações dos experimentos e nas execuções das técnicas. Além disso, os custos associados para execuções das ferramentas (Browserbite é uma ferramenta comercial, ou seja, seria necessário um investimento financeiro) inviabilizaram uma avaliação de comparação direta. Futuros estudos precisam ser conduzidos para comparar a precisão entre estas abordagens.

Outra limitação à validade do experimento está associada as aplicações Web que foram usadas para construir o conjunto de dados. As aplicações Web foram implementadas por desenvolvedores iniciantes, que podem ter aumentado o número e a variedade de XBIs que foram analisados. As aplicações Web replicaram o leiaute de sites brasileiros de alto tráfego, apresentando complexidade de estrutura DOM similar as aplicações Web do mundo real. Eles foram implementados utilizando apenas HTML e CSS, sem o uso de JavaScript para fornecer comportamento dinâmico no DOM, eventos de temporização ou elementos animados (usando a transição CSS3 ou o elemento HTML5 de video). As mudanças dinâmicas na estrutura DOM impacta principalmente a capacidade da abordagem proposta de combinar o mesmo elemento DOM renderizado em diferentes navegadores. No entanto, isso não invalida o estudo apresentado, visto que o foco foi identificar XBIs de Leiaute. Choudhary et al. (2013, 2014) relataram uma abordagem heurística para realizar esta tarefa de correspondência de DOM. Em trabalhos futuros, esta heurística pode ser integrada para incluir aplicações Web que contenham comportamento dinâmico na estrutura DOM.

No que concerne ao desempenho da abordagem de detecção XBI, observou-se gargalo na etapa da aquisição de dados, devido a necessidade de navegar por todos os elementos DOM da página Web em cada navegador avaliado. O site mais complexo do experimento, apresentou 280 elementos DOM e o coletor Web demorou algumas horas para completar a extração das características dos elementos. Este fato, destaca a necessidade de otimizar o processo de

coleta e motivam trabalhos futuros. Uma possibilidade para reduzir o tempo de análise das aplicações Web complexas, seria selecionar apenas os elementos DOM que são mais propensos em apresentar XBIs, sem prejudicar o percentual de identificação do conjunto completo de XBIs existentes.

3.5 DISCUSSÃO

O presente capítulo reportou a modelagem de uma abordagem de detecção de XBI de Leiaute com base nas técnicas de classificação. Na abordagem proposta, as aplicações Web foram segmentadas utilizando sua estrutura DOM e as características usadas no modelo de classificação foram sintetizadas com base na posição, tamanho e imagem dos elementos DOM. O experimento apresentou evidências que corroboraram com a hipótese de que a abordagem proposta poderia identificar automaticamente os XBIs em um grupo específico de aplicações Web. O melhor desempenho do classificador foi obtido por uma árvore de decisão C5.0 com *F-measure* de 0.91, configurada com o atributo *winnowing* e o algoritmo de agrupamento com 20 iterações *boosting*. Assim, o experimento apresentou indicadores que podem ser generalizado e usados em outras aplicações Web para melhorar o processo de engenharia Web.

Choudhary et al. (2013, 2014) relataram o desenvolvimento de uma abordagem similar na ferramenta X-Pert, segmentando também a aplicação Web de acordo com sua estrutura DOM. No entanto, além de utilizar comparação da posição e da imagem para determinar XBIs, os seguintes aspectos discernem desta abordagem: Eles consideram a posição relativa dos elementos DOM, o *script crawler* utilizado não tem a capacidade de ocultar os elementos DOM após capturar as propriedades, e reportaram no critério de avaliação de XBIs a utilização da comparação da imagem somente no nó folha da árvore DOM. Este critério de avaliação foi usado para que as diferenças de apresentação observadas na imagem de um elemento (nó filho) não propagasse para análise do elemento pai (CHOUDHARY et al., 2013). Por outro lado, na abordagem proposta usou um processo de coleta mais elaborado, no qual as propriedades do elemento DOM foram coletadas em uma ordem específica (dos níveis mais baixos da árvore para os níveis elevados da árvore) e depois de coletada as propriedades do elemento DOM, o mesmo era ocultado. Portanto, a comparação de imagens foi usada para determinar XBIs em todos os elementos, nesta abordagem. Choudhary et al. (2013) também apresentaram uma abordagem para detectar os XBIs em funcionalidades, enquanto esta pesquisa enfatizou especificamente em XBI de Leiaute.

Outra abordagem de detecção de XBI nos trabalhos relacionados foi o Browserbite(SAAR et al., 2016). Browserbite, diferentemente da abordagem proposta, segmenta as apli-

cações Web em Regiões de Interesse e modela um sistema de classificação utilizando características de imagem para identificar XBIs. O uso da estrutura DOM para segmentar aplicações Web aumenta o custo de coleta, já que cada elemento DOM da estrutura deve ser analisado separadamente. Por outro lado, uma análise separada dos elementos DOM possivelmente melhora o *feedback* para dos usuários, uma vez que as abordagens de detecção de XBI da estrutura DOM podem reportar o elemento DOM exato que apresentou um erro.

Em relação ao modelo de classificação, o conjunto de características do Browserbite inclui o navegador que está sendo testado, de modo a criar um arquétipo de aprendizado de máquina que poderá apresentar resultados diferentes dependendo do navegador testado. Esta estratégia foi adotada, considerando que alguns navegadores apresentam tendências distintas de XBIs. Portanto, para testar uma versão específica de navegador, um conjunto de dados de aprendizagem deve ser anteriormente construído para que possa ser analisado corretamente. Porém, neste projeto de pesquisa, o modelo de classificação foi construído com um conjunto de características independente de navegador, possibilitando que todos navegadores sejam avaliados de forma semelhante. Sendo assim, a inclusão de versões de navegadores mais recentes no cenário de avaliação não requer a construção de um novo conjunto de dados de aprendizado. Na abordagem proposta, este cenário de avaliação poderá ser aplicado em outras plataformas de navegador, por exemplo, *Smartphones* e *Tablets*.

A Tabela 3.3 foram destacados os diferenciais entre a abordagem proposta em relação aos trabalhos relacionados considerados estado da arte.

Tabela 3.3: Diferenciais entre a abordagem proposta em relação aos trabalhos relacionados.

Abordagem Proposta	**X-Pert (CHOUDHARY et al., 2013)**	**Browserbite (SAAR et al., 2016)**
Abordagem baseada na árvore DOM e visão computacional.	Abordagem baseada na árvore DOM e visão computacional.	Abordagem baseada em visão computacional.
Esconde elemento DOM após aquisição das propriedades.	Não esconder elementos DOM.	Não utiliza árvore DOM.
Critério de avaliação compara imagem de todos os elementos coletados da árvore DOM.	Compara imagem apenas do nó folha da árvore DOM.	Não utiliza árvore DOM, realiza a segmentação de regiões de interesse através de características da imagem.
Abordagem detecta XBI de Leiuate.	Abordagem detecta XBI de Leiaute e Funcionalidade.	Abordagem detecta XBI de Leiaute.
Modelo de classificação construído com um conjunto de características independente de navegador.	Não depende do navegador como característica no modelo de classificação.	Dependência do navegador como característica no modelo de classificação.
Utiliza árvore de decisão (algoritmo C5.0).	Utiliza árvore de decisão.	Utiliza redes neurais.

Neste estudo, o classificador que apresentou os melhores resultados (*F-measure*=0,91), reportou *Precision* de 0,94 e *Recall* de 0,89. O menor valor apresentado pela métrica *Recall*

está associado ao número de falsos negativos. São os XBIs existentes nas aplicações Web que não foram identificados pela abordagem proposta. Ao analisar os falsos negativos da técnica proposta, foi identificado que a maioria deles estão associados ao XBIs de Leiaute Interno. A abordagem utilizou para detectar XBIs de Leiaute Interno duas características principais: *Image Diff* e $\chi 2$. Ambas as características apresentaram um valor numérico que representa diferenças entre duas imagens, comparando o navegador referência e o navegador alvo. No entanto, houve casos de comparação de imagem em que ambas as métricas não foram sensíveis o suficiente para dar origem a um XBI. Por exemplo, mudanças na cor dos textos que representam um XBI, foram medidas de forma semelhante à posição de fundo da imagem, resultando em perdas imperceptíveis que não representaram XBIs. Portanto, as características de processamento de imagem, como as usadas no Browserbite (SAAR et al., 2016), em trabalhos futuros podem melhorar os resultados reportados neste experimento, possivelmente aumentando a métrica *Recall* dos classificadores.

3.6 CONSIDERAÇÕES FINAIS

Neste capítulo, foi proposta uma abordagem para detecção automática de XBI de Leiaute, baseada no uso de aprendizado de máquina, comparação de imagens e segmentação da árvore DOM que compõe uma aplicação Web. O objetivo principal era avaliar se o modelo de classificação poderia detectar automaticamente XBIs de Leiaute em aplicações Web. Foi considerado no conjunto de características do modelo de classificação as diferenças de posição e tamanho para detectar os XBIs externos, e a comparação de imagem (*screenshot*) para identificar os XBIs internos. O modelo de classificação foi validado usando algoritmos de árvore de decisão. Além disso, nesta abordagem foi considerada uma estratégia de varredura de elementos DOM distinta, seguindo uma metodologia *bottom-up*, coletando dados dos elementos de nível mais baixo da árvore e subindo para os níveis superiores da árvore DOM. Na estratégia de varredura, também foi removida a visibilidade dos elementos DOM depois que eles foram inspecionados. Está estratégia foi conduzida para aumentar a eficácia da abordagem, quando comparada a trabalhos relacionados. Segundo os resultados obtidos no experimento, o modelo de classificação alcançou o valor 0,91 na métrica *F-measure* através do procedimento *Cross-Validation*. Desta forma, refletem evidências que suportam a hipótese de que a abordagem proposta pode ser utilizada para identificação de XBIs de Leiaute em aplicações Web.

No próximo capítulo, conclui-se esta dissertação e, objetivamente, as principais contribuições e as limitações revisitadas. Trabalhos futuros serão apresentados, delineando possíveis tópicos de interesse para novas pesquisas.

4 CONCLUSÕES

Atualmente, as aplicações Web tornaram-se mais ricas e interativas (WATANABE et al., 2015) em razão da “Web 2.0” and HTML5. Apesar disso, os recentes avanços em termo de tecnologia elevaram a complexidade, aumentaram os custos e esforços gastos no processo de desenvolvimento de *software*, uma vez que as aplicações Web precisam suportar múltiplos navegadores.

O impacto de XBIs pode variar de inconsistências insignificantes a falhas críticas no Leiaute ou na funcionalidade de aplicações Web. Desta forma, os XBIs podem afetar diretamente a experiência do usuário enquanto navega na aplicação Web. O problema de teste manual de cross-browser está associado a grande variedade de navegadores e sistemas operacionais. O testador deve inspecionar manualmente essas aplicação em diferentes navegadores para garantir se o comportamento está consistente. Portanto, esta tarefa é lenta e requer muita atenção do profissional.

Nesse contexto, o principal objetivo desta dissertação era propor uma abordagem para detecção automática de XBI de Leiaute. A tarefa de detecção automática de XBI foi fundamentada como problema de classificação, sendo assim, utilizou-se de aprendizado de máquina, segmentação da árvore DOM e comparação de imagens. Foi conduzido um experimento que investigou a eficácia do modelo de classificação em detectar de forma automática os XBIs de Leiaute. O modelo de classificação foi construído através de um conjunto de características baseadas nas diferenças de posição, tamanho e comparação das imagens do elemento DOM. Por fim, o modelo foi validado por meio de algoritmos de árvore de decisão. Os resultados obtidos forneceram evidências positivas em que a abordagem proposta poderia identificar automaticamente XBIs em aplicações Web e a abordagem alcançou eficácia semelhante ao relatado pela ferramenta Browserbite.

As contribuições desta dissertação são sumarizadas na Seção 4.1. Na Seção 4.2 são apresentadas as limitações e trabalhos futuros que fomentam a continuação de pesquisas no tema.

4.1 CONTRIBUIÇÕES

Nesta seção são revisitadas as principais contribuições identificadas na condução da dissertação.

Levantamento de abordagens de detecção automática de XBIs. No Capítulo 2 foi

conduzido um processo de revisão sistemática da literatura. Como resultado, este processo contribui com a identificação das diferentes abordagens utilizadas para detecção automática de XBIs, consideradas o estado da arte. Foi possível levantar a quantidade de artigos existentes sobre o assunto, os pesquisadores, as técnicas mais relevantes e principalmente as limitações existentes nos estudos. Esta revisão sistemática forneceu uma base para condução dos próximos passos deste projeto de pesquisa.

Elaboração de processo de segmentar e ocultar os elementos na árvore DOM. Com a finalidade de armazenar as propriedades do elemento HTML, um algoritmo foi desenvolvido para realizar o processo de varredura da árvore DOM seguindo uma metodologia *bottom-up* (Inicia a coleta das propriedades do elemento de nível mais baixo e sobe para os níveis mais elevados da árvore DOM). O processo de varredura inicia pelos nós folhas extraindo as propriedades (posição, tamanho e *screenshot*), em seguida, oculta o elemento da árvore DOM, por fim, sobe para o próximo nível. Esta estratégia contribui para que os possíveis XBIs associados a um elemento (nó folha) não propague ou impacte a análise de seu nó pai na estrutura da árvore DOM. O código desenvolvido nesta dissertação foi disponível com licença de software livre e encontram-se disponíveis em repositórios online.

Elaboração de uma estratégia de detecção de XBIs de Leiaute Internos e Externos. Com objetivo de aumentar a efetividade na detecção automática de XBIs, foi elaborada uma estratégia de segregar em duas categorias as inconsistências relacionadas ao Leiaute. A estratégia contribuiu para definição das características que foram utilizadas para modelar o problema de classificação. Como resultado, foi possível analisar que os falsos negativos apresentados pela abordagem estão associados ao XBI Interno em sua maioria.

4.2 LIMITAÇÕES E TRABALHOS FUTUROS

Comparação direta da eficácia da abordagem proposta em relação aos trabalhos relacionados. O experimento realizado nesta pesquisa apresentou evidências que poderiam ser generalizado e usadas em outras aplicações Web. A execução do processo apresentou nível de efetividade semelhante ao relatado na ferramenta Browserbite. Apesar disso, as respostas obtidas em ambos os estudos não podem ser comparadas, devido as diferenças nas configurações e execuções do experimentos. Futuros estudos poderão ser conduzidos para comparar a eficácia entre estas abordagens.

Conjunto de dados utilizado no experimento. As aplicações Web que constituem o conjunto de dados, foram construídas por desenvolvedores iniciantes, o que pode contribuir

para um número ampliado de XBIs. As aplicações Web foram implementados sem o uso de JavaScript para fornecer comportamento dinâmico no DOM. As mudanças dinâmicas na estrutura DOM impactariam principalmente a capacidade da abordagem proposta de combinar o mesmo elemento DOM renderizado em diferentes navegadores, no entanto, não invalida o estudo apresentado, visto que o foco foi identificar XBIs de Leiaute. Em trabalhos futuros poderá ser integrada uma heurística que realize a tarefa de correspondência de DOM das aplicações Web do mundo real que contém comportamento dinâmico na estrutura DOM (CHOUDHARY et al., 2013, 2014).

Refatoração da etapa de aquisição de dados para melhoria do desempenho. Foi constatado um ponto crítico na etapa de aquisição de dados, devido a necessidade de navegar por todos os elementos DOM da página Web em cada navegador avaliado. O procedimento demorou algumas horas para completar a extração de dados em aplicações Web mais complexas. Uma possibilidade para reduzir o tempo de coleta, seria selecionar apenas os elementos DOM que são mais vulneráveis em apresentar XBIs, sem prejudicar o percentual de identificação do conjunto completo de XBIs existentes.

Falsos Negativos. Foi identificado que a maioria dos falsos negativos apresentados pela abordagem estão associados ao XBIs de Leiaute Interno. A abordagem utilizou técnicas de comparação de imagem, onde houve casos em que não foram sensíveis o suficiente para apontar um XBI. Sendo assim, características de processamento de imagem usadas no Browserbite (SAAR et al., 2016), poderão ser experimentadas em trabalhos futuros.

Avaliação do modelo de classificação em dispositivos móveis. A abordagem proposta poderá ser estendida para incorporar outros navegadores (*Safari, Microsoft Edge, Opera*) e também plataformas de navegadores (*smartphones*, *tablets*).

4.3 DIVULGAÇÃO DOS RESULTADOS

Ao longo do desenvolvimento da dissertação de mestrado, os resultados foram publicados nos seguintes trabalhos:

- Artigo publicado na revista JADI 2016 - *Journal on Advances in Theoretical and Applied Informatics*: PAES, F. C.; WATANABE, W. M. . **Detecção Automática de Incompatibilidades Cross-Browser utilizando Redes Neurais Artificiais**. Journal on Advances in Theoretical and Applied Informatics, v. 2, p. 55-59, 2016.;
- Artigo publicado no SAC 2017 - *Symposium on Applied Computing*: PAES, F. C. . **Auto-**

matic detection of cross-browser incompatibilities using machine learning and screenshot similarity. In: the Symposium, 2017, Marrakech. Proceedings of the Symposium on Applied Computing - SAC '17. New York: ACM Press, 2017. p. 697.;

- Artigo publicado no CIBSE 2017 - *Ibero-American Conference on Software Engineering*: PAES, F. C.; WATANABE, W. M. . **Automatic Detection of Cross-Browser Incompatibilities: Machine Learning Techniques and Screenshot Similarity**. In: Ibero-American Conference on Software Engineering, 2017, Buenos Aires. Ibero-American Conference on Software Engineering.;

- Artigo aceito para publicação no SAC 2018 - *Symposium on Applied Computing*: PAES, F. C.; WATANABE, W. M. .**Layout Cross-Browser Incompatibility Detection using Machine Learning and DOM Segmentation**. In: the Symposium, 2018, Pau. Proceedings of the Symposium on Applied Computing - SAC '18.

Os seguintes trabalhos de pesquisa encontram-se em preparação:

- Artigo a ser submetido em revista na área de engenharia de software. WATANABE, W. M.; PAES, F. C. **Towards Cross-Browser Incompatibilities Detection: a Systematic Literature Review**

- Artigo a ser submetido em revista na área de engenharia de software. WATANABE, W. M.; PAES, F. C. **Layout Cross-Platform and Cross-Browser Incompatibilities Detection using Classification of DOM elements**

REFERÊNCIAS

AHMAD, R.; LI, Z.; AZAM, F. Web engineering: a new emerging discipline. In: **Proceedings of the IEEE Symposium on Emerging Technologies, 2005.** 2005. p. 445–450.

BADAMPUDI, D.; WOHLIN, C.; PETERSEN, K. Experiences from using snowballing and database searches in systematic literature studies. In: **Proc. of the 19th International Conference on Evaluation and Assessment in Software Engineering (EASE 2015)**. New York, NY, USA: ACM, 2015. p. 17:1–17:10. ISBN 978-1-4503-3350-4. Disponível em: <http://doi.acm.org/10.1145/2745802.2745818>.

BRERETON, P. et al. Lessons from applying the systematic literature review process within the software engineering domain. In: . New York, NY, USA: Elsevier Science Inc., 2007. v. 80, n. 4, p. 571–583. ISSN 0164-1212. Disponível em: <http://dx.doi.org/10.1016/j.jss.2006.07.009>.

BUCHANAN, B. G.; SHORTLIFFE, E. H.; MELLE, W. van. Emycin: A knowledge engineer's tool for constructing rule-based expert systems. In: ____. **Rule-based expert systems**: The mycin experiments of the stanford heuristic programming project. Addison-Wesley, 1984. cap. 15, p. 303–313. ISBN 0201101726. Disponível em: <http://people.dbmi.columbia.edu/ ehs7001/Buchanan-Shortliffe-1984/Chapter-15.pdf>.

BUDGEN, D.; BRERETON, P. Performing systematic literature reviews in software engineering. In: **Proc. of the 28th International Conference on Software Engineering (ICSE 2006)**. New York, NY, USA: ACM, 2006. p. 1051–1052. ISBN 1-59593-375-1. Disponível em: <http://doi.acm.org/10.1145/1134285.1134500>.

CHEN, T. Y. et al. Adaptive random testing: The art of test case diversity. **Journal of Systems and Software**, Elsevier Science Inc., New York, NY, USA, v. 83, n. 1, p. 60–66, jan. 2010. ISSN 0164-1212. Disponível em: <http://dx.doi.org/10.1016/j.jss.2009.02.022>.

CHOUDHARY, S. Detecting cross-browser issues in web applications. In: **Proc. of the 33rd International Conference on Software Engineering (ICSE 2011)**. 2011. p. 1146–1148. ISSN 0270-5257.

CHOUDHARY, S.; PRASAD, M.; ORSO, A. Crosscheck: Combining crawling and differencing to better detect cross-browser incompatibilities in web applications. In: **Proc. of the 5thInternational Conference on Software Testing, Verification and Validation (ICST 2012)**. 2012. p. 171–180.

CHOUDHARY, S.; PRASAD, M.; ORSO, A. X-pert: Accurate identification of cross-browser issues in web applications. In: **Proc. of the 35th International Conference on Software Engineering (ICSE 2013)**. 2013. p. 702–711.

CHOUDHARY, S.; VERSEE, H.; ORSO, A. A cross-browser web application testing tool. In: **Proc. of the 2010 IEEE International Conference on Software Maintenance (ICSM 2010)**. 2010. p. 1–6. ISSN 1063-6773.

CHOUDHARY, S.; VERSEE, H.; ORSO, A. Webdiff: Automated identification of cross-browser issues in web applications. In: **Proc. of the 2010 IEEE International Conference on Software Maintenance (ICSM 2010)**. 2010. p. 1–10. ISSN 1063-6773.

CHOUDHARY, S. R.; PRASAD, M. R.; ORSO, A. X-pert: A web application testing tool for cross-browser inconsistency detection. In: **Proc. of the 2014 International Symposium on Software Testing and Analysis (ISSTA 2014)**. New York, NY, USA: ACM, 2014. p. 417–420. ISBN 978-1-4503-2645-2. Disponível em: <http://doi.acm.org/10.1145/2610384.2628057>.

DALLMEIER, V. et al. Webmate: A tool for testing web 2.0 applications. In: **Proc. of the Workshop on JavaScript Tools (JSTools 2012)**. New York, NY, USA: ACM, 2012. p. 11–15. ISBN 978-1-4503-1274-5. Disponível em: <http://doi.acm.org/10.1145/2307720.2307722>.

DALLMEIER, V. et al. Webmate: Generating test cases for web 2.0. In: **Software Quality. Increasing Value in Software and Systems Development: 5th International Conference, SWQD 2013, Vienna, Austria, January 15-17, 2013. Proceedings**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 55–69. ISBN 978-3-642-35702-2.

DALLMEIER, V. et al. Webmate: Web application test generation in the real world. In: **Proc. of the Workshops of the IEEE 7th International Conference on Software Testing, Verification and Validation Workshops (ICSTW 2014)**. 2014. p. 413–418.

HEATON, J. **Introduction to Neural Networks for Java, 2Nd Edition**. 2nd. ed. : Heaton Research, Inc., 2008. ISBN 1604390085, 9781604390087.

KITCHENHAM, B. **Procedures for performing systematic reviews**. Keele, UK, 2004. 1–26 p.

LI, X.; ZENG, H. Modeling web application for cross-browser compatibility testing. In: **Proc. of the 15th IEEE/ACIS International Conference on Software Engineering (ICSE 2014), Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2014)**. 2014. p. 1–5.

MESBAH, A.; DEURSEN, A. van. Invariant-based automatic testing of ajax user interfaces. In: **Proc. of the 31st International Conference on Software Engineering (ICSE 2009)**. Washington, DC, USA: IEEE Computer Society, 2009. p. 210–220. ISBN 978-1-4244-3453-4. Disponível em: <http://dx.doi.org/10.1109/ICSE.2009.5070522>.

MESBAH, A.; DEURSEN, A. van; LENSELINK, S. Crawling ajax-based web applications through dynamic analysis of user interface state changes. **ACM Transactions on the Web**, ACM, New York, NY, USA, v. 6, n. 1, p. 3:1–3:30, mar. 2012. ISSN 1559-1131. Disponível em: <http://doi.acm.org/10.1145/2109205.2109208>.

MESBAH, A.; PRASAD, M. R. Automated cross-browser compatibility testing. In: **Proceedings of the 33rd International Conference on Software Engineering (ICSE 2011)**. New York, NY, USA: ACM, 2011. p. 561–570. ISBN 978-1-4503-0445-0. Disponível em: <http://doi.acm.org/10.1145/1985793.1985870>.

PAES, F. C.; WATANABE, W. M. Layout cross-browser incompatibility detection using machine learning and dom segmentation. In: **Proceedings of the Symposium on Applied Computing**. New York, NY, USA: ACM, 2018. (SAC '18).

PELE, O.; WERMAN, M. The quadratic-chi histogram distance family. In: **Proc. of the 11th European Conference on Computer Vision: Part II (ECCV 2010)**. Berlin, Heidelberg: Springer-Verlag, 2010. (ECCV'10), p. 749–762. ISBN 3-642-15551-0, 978-3-642-15551-2. Disponível em: <http://dl.acm.org/citation.cfm?id=1888028.1888085>.

POWERS, D. M. W. Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. **Journal of Machine Learning Technologies**, v. 2, n. 1, p. 37–63, 2011.

QUINLAN, J. R. Induction of decision trees. **Mach. Learn.**, Kluwer Academic Publishers, Hingham, MA, USA, v. 1, n. 1, p. 81–106, mar. 1986. ISSN 0885-6125. Disponível em: <http://dx.doi.org/10.1023/A:1022643204877>.

QUINLAN, J. R. **C4.5: Programs for Machine Learning**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. ISBN 1-55860-238-0.

ROKACH, L.; MAIMON, O. Decision trees. In: ____. **Data Mining and Knowledge Discovery Handbook**. Boston, MA: Springer US, 2005. cap. 9, p. 165–192. ISBN 978-0-387-25465-4. Disponível em: <https://doi.org/10.1007/0-387-25465-X_9>.

RUBNER, Y.; TOMASI, C.; GUIBAS, L. J. The earth mover's distance as a metric for image retrieval. **International Journal of Computer Vision**, Kluwer Academic Publishers, Hingham, MA, USA, v. 40, n. 2, p. 99–121, nov. 2000. ISSN 0920-5691. Disponível em: <http://dx.doi.org/10.1023/A:1026543900054>.

SAAR, T. et al. Cross-browser testing in browserbite. In: ____. **Proceedings of the 14th International Conference Web Engineering (ICWE 2014)**. Cham: Springer International Publishing, 2014. p. 503–506. ISBN 978-3-319-08245-5.

SAAR, T. o. et al. Browserbite: Cross-browser testing via image processing. **Softw. Pract. Exper.**, John Wiley & Sons, Inc., New York, NY, USA, v. 46, n. 11, p. 1459–1477, nov. 2016. ISSN 0038-0644. Disponível em: <https://doi.org/10.1002/spe.2387>.

SANCHEZ, L.; JR., P. T. A. Automatic deformations detection in internet interfaces: Addii. In: **Proceedings of the 17th International Conference on Human-Computer Interaction. Part III: Users and Contexts (HCI International 2015)**. : Springer International Publishing, 2015. (Lecture Notes in Computer Science, v. 9171), p. 43–53.

SELAY, E.; ZHOU, Z. Q.; ZOU, J. Adaptive random testing for image comparison in regression web testing. In: **Proc. of the 2014 International Conference on Digital Image Computing: Techniques and Applications (DICTA 2014)**. 2014. p. 1–7.

SEMENENKO, N. **Accurate diagnosis of cross-browser compatibility issues via machine learning**. Tese (Master's Thesis) — University of Tartu, Tartu, Estonia, 2013.

SEMENENKO, N.; DUMAS, M.; SAAR, T. Browserbite: Accurate cross-browser testing via machine learning over image features. In: **Proc. of the 29th IEEE International Conference on Software Maintenance (ICSM 2013)**. 2013. p. 528–531. ISSN 1063-6773.

SHI, H.; ZENG, H. Cross-browser compatibility testing based on model comparison. In: **Computer Application Technologies (CCATS), 2015 International Conference on**. 2015. p. 103–107.

WATANABE, W. M.; GERALDO, R. J.; FORTES, R. P. de M. Keyboard navigation mechanisms in widgets: an investigation on ARIA's implementations. **Journal of Web Engineering**, v. 14, n. 1&2, p. 41–62, 2015.

WEISS, S. M.; KULIKOWSKI, C. A. **Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1991. ISBN 1-55860-065-5.

XU, S.; ZENG, H. Static analysis technique of cross-browser compatibility detecting. In: **Applied Computing and Information Technology/2nd International Conference on Computational Science and Intelligence (ACIT-CSI), 2015 3rd International Conference on**. 2015. p. 103–107.